MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

OpenLDAP Directory Services in FreeBSD

Fluentd-UI and Suricata IDS

FreeBSD, Google Cloud, and Dual ECC/RSA

Mongoose Embedded Web Server on FreeBSD

Using PostgreSQL Foreign Data Wrapper to Keep Track of Files

Free RDP Configuration

VOL 11 NO 11
ISSUE 11/2017 (99)
ISSN 1898-9144

IS AFFORDABLE FLASH STORAGE OUT OF REACH?

NOT ANYMORE!

IXSYSTEMS DELIVERS A FLASH ARRAY FOR UNDER $10,000.

Introducing FreeNAS® Certified Flash: A high performance all-flash array at the cost of spinning disk.

- Unifies NAS, SAN, and object storage to support multiple workloads
- Runs FreeNAS, the world's #1 software-defined storage solution
- Performance-oriented design provides maximum throughput/IOPs and lowest latency
- OpenZFS ensures data integrity
- Perfectly suited for Virtualization, Databases, Analytics, HPC, and M&E
- 10TB of all-flash storage for less than $10,000
- Maximizes ROI via high-density SSD technology and inline data reduction
- Scales to 100TB in a 2U form factor

The all-flash datacenter is now within reach. Deploy a FreeNAS Certified Flash array today from iXsystems and take advantage of all the benefits flash delivers.

Call or click today! **1-855-GREP-4-IX** (US) **|** **1-408-943-4100** (Non-US) **|** **www.iXsystems.com/FreeNAS-certified-servers**

DON'T DEPEND ON CONSUMER-GRADE STORAGE.

intel inside™

KEEP YOUR DATA SAFE!

Buy from amazon.com

USE AN ENTERPRISE-GRADE STORAGE SYSTEM FROM IXSYSTEMS INSTEAD.

The FreeNAS Mini: Plug it in and boot it up — it just works.

- Runs FreeNAS, the world's #1 software-defined storage solution
- Unifies NAS, SAN, and object storage to support multiple workloads
- Encrypt data at rest or in flight using an 8-Core 2.4GHz Intel® Atom® processor
- OpenZFS ensures data integrity
- A 4-bay or 8-bay desktop storage array that scales to 48TB and packs a wallop
- Backed by a 1 year parts and labor warranty, and supported by the Silicon Valley team that designed and built it
- Perfectly suited for SoHo/SMB workloads like backups, replication, and file sharing
- Lowers storage TCO through its use of enterprise-class hardware, ECC RAM, optional flash, white-glove support, and enterprise hard drives

And really — why would you trust storage from anyone else?

iXsystems™

Call or click today! **1-855-GREP-4-IX** (US) **|** **1-408-943-4100** (Non-US) **|** **www.iXsystems.com/Freenas-Mini** or purchase on Amazon.

EDITOR'S WORD

MAGAZINE BSD

Dear Readers,

I hope you are well. The editor's word will sound a little bit different since we are close to ushering the final edition of the year. It's a month that we reflect on our lives, our goals, and most importantly, plans on what to do in the following year. I hope you enjoyed all the monthly issues of 2017. Suffice to say, there will be one more issue at the end of December just to crown this wonderful year of 2017. We really appreciate your readership and engagement, and hope to provide you with more meaningful content in 2018.

I hope you started doing your New Year's lists and Christmas presents lists. I believe it's that time of the year when we should have some fun and joy by providing surprises for others. I really like Christmas time and the mood it sets towards the end of the year. I hope that you will have time to prepare and eventually enjoy the good moments during this festive season before we come to the close of the year.

I also hope that you have prepared a list on what you would like to learn in 2018. Which interesting tool and technology have you found lately during your free time? Please share them with me so that I can surprise you with an online course on this tool or technology in 2018. With BSD magazine, be sure to learn about your favourite tool or technology. This is one of our obligations to our readers. Hence, I look forward to your emails.

Now, let's take a peek into this issue. You will find many great and technically interesting articles for you. I really like all the articles, and I am thankful to all the authors for their patience when we were preparing them. I invite you to look over the list of articles on the next page. Lastly, a big thank you to all our reviewers for their valuable suggestions on how to make the articles better.

So, just sit back, get a light drink, and engage with the authors' minds.

Enjoy reading,

Ewa & The BSD Team
ewa@bsdmag.org

Note!

Remember to read TOOL REVIEW on page 6 and learn about ***Web Development Forensics with BugReplay by David Carlier***

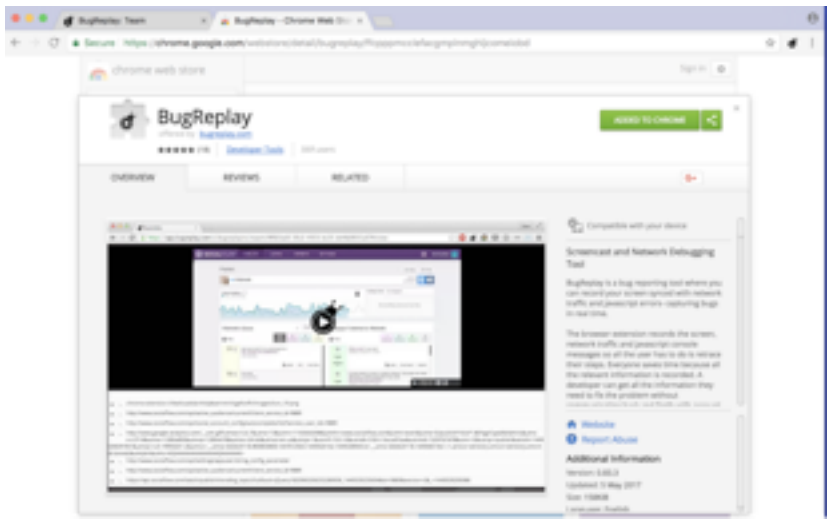
TABLE OF CONTENTS

TOOL REVIEW

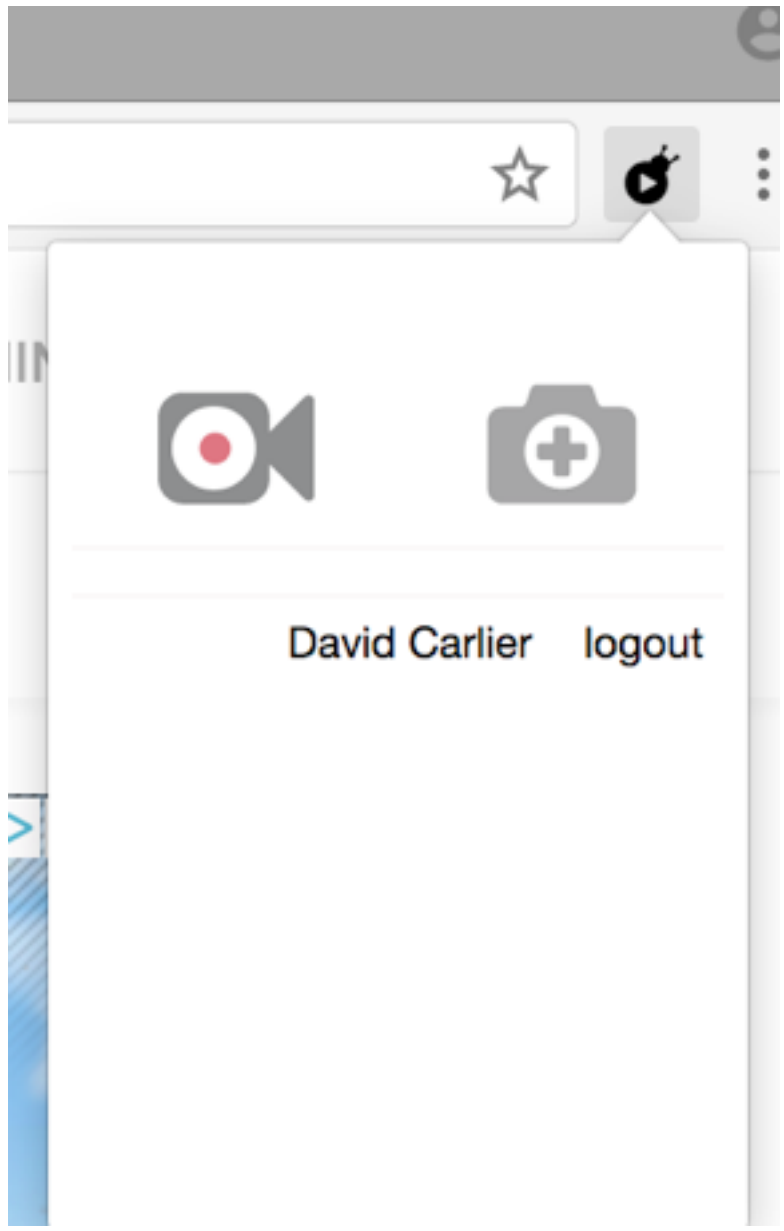
Web Development Forensics with BugReplay

reviewed by David Carlier

Every web developer has his set of tools for debugging a web application. The most common developer tools are Firebug for Firefox, Chrome's Developer Tools, and Internet Explorer's Developer Toolbar. All of these can be used to inspect HTML elements, Javascript, CSS styles, network traffic, and Ajax calls. These tools are great, but have one downside: all of these data is produced in real time but not recorded. But what if you want to spot a specific bug scenario to study and to replay it at will? To display to the rest of the development team, your client, etc. This can be useful especially if the bug is triggered only in a very specific case such as a web form filled with invalid data, triggering an Ajax call which takes longer than it should, triggering a bug in client-side Javascript. We can go on a long list of possible use cases when this type of tool. To use BugReplay, you'll need the Chrome/Chromium/Iridium browser; you'll also need to download the extension and register an account on http://www.bugreplay.com.

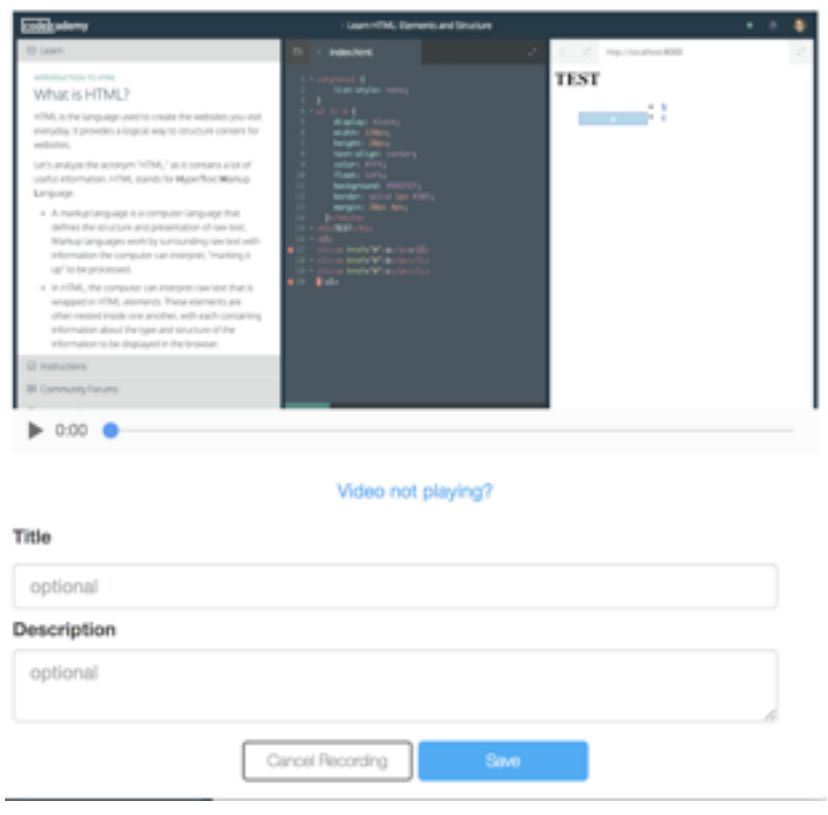


The BugReplay extension works basically in two modes. Snapshot mode takes a static picture of the current page to be able to spot, for example, misplaced HTML elements. Video mode records for a couple of seconds to be able to reproduce a specific scenario as explained above with those two explicit icons.



As you can see in the last illustration, there is a blue icon to allow us to report or to give immediate feedback to BugReplay team within reasonable work hours. Once the snapshot is taken, you can apply various modifications such as cropping, resizing, and putting a comment and drawing on top of it to highlight the problem, useful even for non-developers individuals.

Then, there is the video mode which completes just as well as the snapshot mode,



where also network and Javascript traffic are recorded (seemingly not available in the trial), video relatively immediately available after post processing to us. Nevertheless, this is a set of tools which can have its place as we can realise through the trial;

One warning: the data that is collected is stored in BugReplay's cloud. Because of this, you will need to decide if this is appropriate from your particular application. Apart of this, it gets the job done as promised.

About BugReplay

BugReplay, a provider of an innovative set of web browser tools that make reporting bugs faster and fixing them easier, today announced the availability of its flagship product of the same name as an add-on for the Firefox web browser. A screencast and network debugging tool for web developers and internal software testers, BugReplay enables users to quickly and accurately submit detailed bug reports about web applications. By creating a synchronized screen recording of a user's actions, network traffic, JavaScript logs and other key environmental data, BugReplay reduces the time to complete the task of bug reporting of up to an hour or more to less than a minute.

Founded in 2015, BugReplay is a leading provider of an innovative set of web browser tools that make reporting bugs faster and fixing them easier. Its mission is to develop easy-to-use tools for diagnosing and repairing issues with web applications. Based in New York City, BugReplay's offerings include: BugReplay, a screencast and network debugging tool for web developers and internal software testers; and Feedback by BugReplay, a reporting tool for website users to submit bug reports to customer support teams. For more information, visit http://www.bugreplay.com and follow on Twitter @BugReplay.

FREEBSD

OpenLDAP Directory Services in FreeBSD (I). Dynamic Configuration Fundamentals

What you will learn:

- Installation and configuration methods for OpenLDAP 2.4 under FreeBSD
- Basic foundations of the new LDAP on-line configuration (OLC)
- Hardening LDAPv3 with SASL and TLS/SSL protocols
- Embedding of NIS+/YP into an LDAP server to provide centralized NIS+ support for UNIX computers
- Administration and basic tuning principles for LDAP servers

What you should already know:

- Intermediate UNIX OS background as end-user and administrator
- Some knowledge of UNIX authentication systems and NIS+/YP
- Experience with FreeBSD system package and FreeBSD ports
- Good taste for command-line usage

The main objective of this article is to introduce directory services managed under the LDAP protocol, and to illustrate a new configuration approach known as Online LDAP Configuration (OCL) which was introduced in OpenLDAP v2.3. We will also present a direct application to encapsulating a NIS+/YP centralized user authentication and management schema for an arbitrary number of servers and clients connected to a TCP/IP network. Additionally, we'll show a web-based administration tool that will make administering the OpenLDAP server easier.

An Overview of Directory Services

Directory services are a special type of database storage systems focused on heterogeneous, hierarchical data. In comparison to traditional full-service standalone relational databases management systems, the number of LDAP read operations exceed write operations. The read operations are mainly searches of special data which follow the patterns described by RFC 1558. For this reason, the standard RDBMS approach is abandoned in favor of key-value databases used as a backend.

LDAPv3 introduces some key improvements over its predecessor LDAPv2 such as:

UTF-8 Internationalization Support for foreign languages

Enhanced Security Mechanisms such as SASL for authentication

In LDAP, authentication is required after the initial "bind" operation. LDAPv3 supports three types of authentication: anonymous, simple, and SASL authentication. A client that sends an LDAP request without doing a "bind" is treated as an anonymous client.

Simple authentication consists of sending the LDAP server the fully qualified Distinguished Name (DN) and a clear-text password of the client. The security weakness with this mechanism is that the password can be read by anyone who can access the network. You can use a simple authentication mechanism within an encrypted channel (such as SSL) to avert exposing the password in this manner provided that it is supported by the LDAP server.

Finally, SASL is the Simple Authentication and Security Layer described by RFC 2222. It specifies a challenge-response protocol in which data is exchanged between the client and the server for authentication and establishment of a security layer on which to carry out subsequent communication. By using SASL, LDAP can support any authentication by a secured negotiation between the LDAP client and server.

Getting Started with OpenLDAP

Installation Procedure

One obvious and important requirement is an updated and running installation of FreeBSD OS, which at the time of writing was FreeBSD 11.1.

It is possible to take advantage of virtualization technologies to simplify the installation and testing process so long as it has a functional internet connection to perform package downloads or a reachable local repository mirror.

First, let us ensure we are up to date with the current patches and fixes in BASE by running:

```
root@laertes:~ # /usr/sbin/freebsd-update fetch

root@laertes:~ # /usr/sbin/freebsd-update install
```

Next, ensure that our package database is up to date and pkgng-ready by running:

```
root@laertes:~ # pkg upgrade

Updating FreeBSD repository catalogue...

FreeBSD repository is up to date.

All repositories are up to date.

Checking for upgrades (1 candidates): 100%

Processing candidates (1 candidates): 100%

Checking integrity... done (0 conflicting)

Your packages are up to date.
```

Following the update of pkg(1), we will need to install the ports tree to install OpenLDAP+SASL as official pre-built binaries for OpenLDAP. DO NOT include SASL support. If you do not require SASL, you can install using the standard `pkg install` routine vs. the ports. To install the ports tree:

```
root@laertes:~ # portsnap fetch extract

Looking up portsnap.FreeBSD.org mirrors... 6
mirrors found.

Fetching public key from
ec2-eu-west-1.portsnap.freebsd.org... done.

Fetching snapshot tag from
ec2-eu-west-1.portsnap.freebsd.org... done.

Fetching snapshot metadata... done.

Fetching snapshot generated at Tue Oct 10 02:00:56
CEST 2017:

…

/usr/ports/x11/yeahconsole/

/usr/ports/x11/yelp/

/usr/ports/x11/zenity/

Building new INDEX files... done.
```

The download and extraction of `portsnaps` tarball takes a while, a good time for coffee or your preferred beverage.

Once `portsnap` has finished its work, install the **portmaster** utility as a pre-built pkg and continue with building our packages:

```
root@laertes:~ # pkg install portmaster

Updating FreeBSD repository catalogue...

Fetching meta.txz: 100%    940 B   0.9kB/s    00:01
```

```
Fetching packagesite.txz: 100%    6 MiB 175.1kB/s
00:35

Processing entries: 100%

FreeBSD repository update completed. 26972 packages
processed.

All repositories are up to date.

The following 1 package(s) will be affected (of 0
checked):

New packages to be INSTALLED:

      portmaster: 3.17.10

Number of packages to be installed: 1

42 KiB to be downloaded.

Proceed with this action? [y/N]: y

[1/1] Fetching portmaster-3.17.10.txz: 100%   42
KiB  42.6kB/s    00:01

Checking integrity... done (0 conflicting)

[1/1] Installing portmaster-3.17.10...

Extracting portmaster-3.17.10: 100%
```

Although it is not strictly necessary, ensure there is consistency with previous FreeBSD releases by means of the following command:

```
root@laertes:~ # pkg2ng

Converting packages from /var/db/pkg

Analysing shared libraries, this will take a
while...

Checking all packages: 100%
```

Once the ports package system has been successfully installed and updated, switch to the /usr/ports/ directory. The port we want to install is OpenLDAP 2.4 server, and is available as net/openldap24-server/ in the ports directory. Remember to install the OpenLDAP server by compiling the sources with the following options selected prior to starting the compilation process **GSSAPI**, **PPOLICY**, **MEMBEROF**, **DYNLIST**, **DYNGROUP**, **REFINT**, **SHA2**, **SASL**, and **UNIQUE** during the openldap24-server port configuration shown in the dialog by Illustration 1

```
root@laertes:~# cd /usr/ports
root@laertes:/usr/ports# portmaster
net/openldap24-server
```

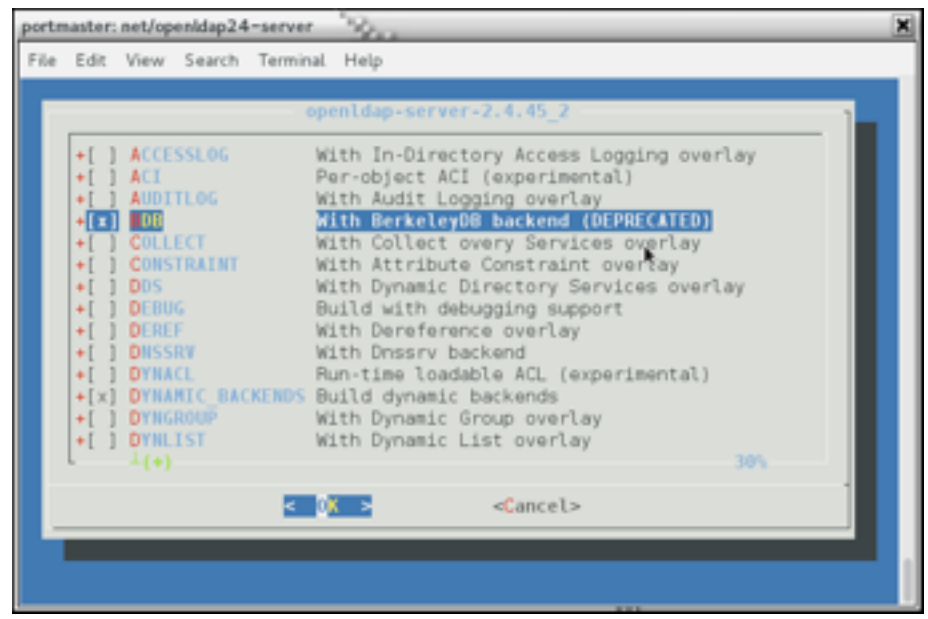


Figure 1: Portmaster OpenLDAP Sever Options Dialog

Additionally, you can choose NLS support for the standard GSSAPI_BASE which is enough for our purposes; as such we need neither HEIMDAL nor MIT support for GSSAPI. The `portmaster` utility will attempt to resolve all of the required dependencies automatically.

```
===>>> The following actions will be taken if you
choose to proceed:

      Install net/openldap24-server

      Install devel/icu

      Install devel/gmake

      Install security/cyrus-sasl2-gssapi

      Install devel/libtool

      Install print/texinfo

      Install devel/gettext-tools

      Install misc/help2man

      Install devel/p5-Locale-gettext

===>>> Proceed? y/n [y]

…

when completed, portmaster will report:

The OpenLDAP server package has been successfully
installed.

In order to run the LDAP server, you need to edit

  /usr/local/etc/openldap/slapd.conf
```

```
to suit your needs and add the following lines to
/etc/rc.conf:

  slapd_enable="YES"

  slapd_flags='-h
"ldapi://%2fvar%2frun%2fopenldap%2fldapi/
ldap://0.0.0.0/"'

slapd_sockets="/var/run/openldap/ldapi"

Then start the server with

  /usr/local/etc/rc.d/slapd start

or reboot.

Try `man slapd' and the online manual at

  http://www.OpenLDAP.org/doc/

for more information.

slapd runs under a non-privileged user id (by
default `ldap'),

see /usr/local/etc/rc.d/slapd for more information.

**************************************************
*********

===>>> Done displaying pkg-message files

===>>> The following actions were performed:

      Installation of devel/gmake (gmake-4.2.1_1)

      Installation of devel/icu (icu-59.1,1)

      Installation of devel/gettext-tools
(gettext-tools-0.19.8.1)

      Installation of devel/p5-Locale-gettext
(p5-Locale-gettext-1.07)

      Installation of misc/help2man
(help2man-1.47.5)

      Installation of print/texinfo (texinfo-6.5,1)

      Installation of devel/libtool (libtool-2.4.6)

      Installation of security/cyrus-sasl2-gssapi
(cyrus-sasl-gssapi-2.1.26_7)

      Installation of net/openldap24-server
(openldap-sasl-server-2.4.45_2)
```

It is recommended to install the gnutls package to enable security features such as TLS/SSL:

```
root@laertes:/usr/ports# portmaster
security/gnutls
```

In addition, select the options available at the dialog in Illustration 2, including UCS4 Unicode Support, NLS, and GSSAPI_BASE option. A long set of packages will be installed to meet all dependencies after a while.

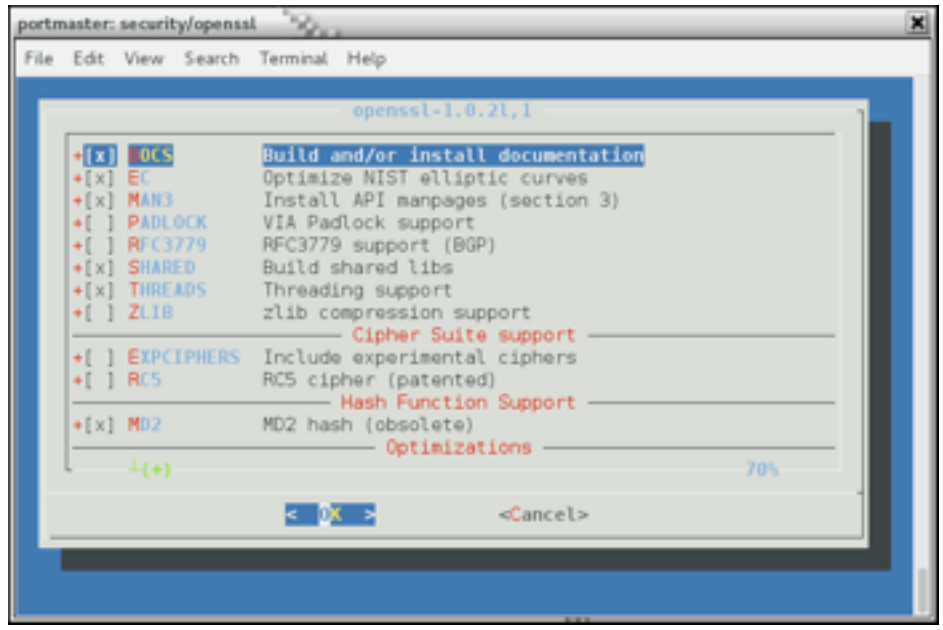


Figure 2: GNUTLS Package Portmaster Installation Dialog

Notice that gnutls installs symlinks to support root certificate discovery by default for software that uses OpenSSL, thereby enabling SSL Certificate Verification by client software without manual intervention. Nevertheless, if you can replace the following symlinks with either an empty file or your site-local certificate bundle if you prefer to do this manually.

/etc/ssl/cert.pem

/usr/local/etc/ssl/cert.pem

/usr/local/openssl/cert.pem

Gnutls utils will be used later on to check some TLS features incorporated into our OpenLDAP server.

TLS/SSL Support

Alternatively, it is possible to use OpenSSL instead of GNUtls. Generally, it is not a good idea to mix packages and ports. Let's use the portmaster method as above.

```
root@laertes:~# cd /usr/ports

root@laertes:/usr/ports# portmaster
security/openssl
```

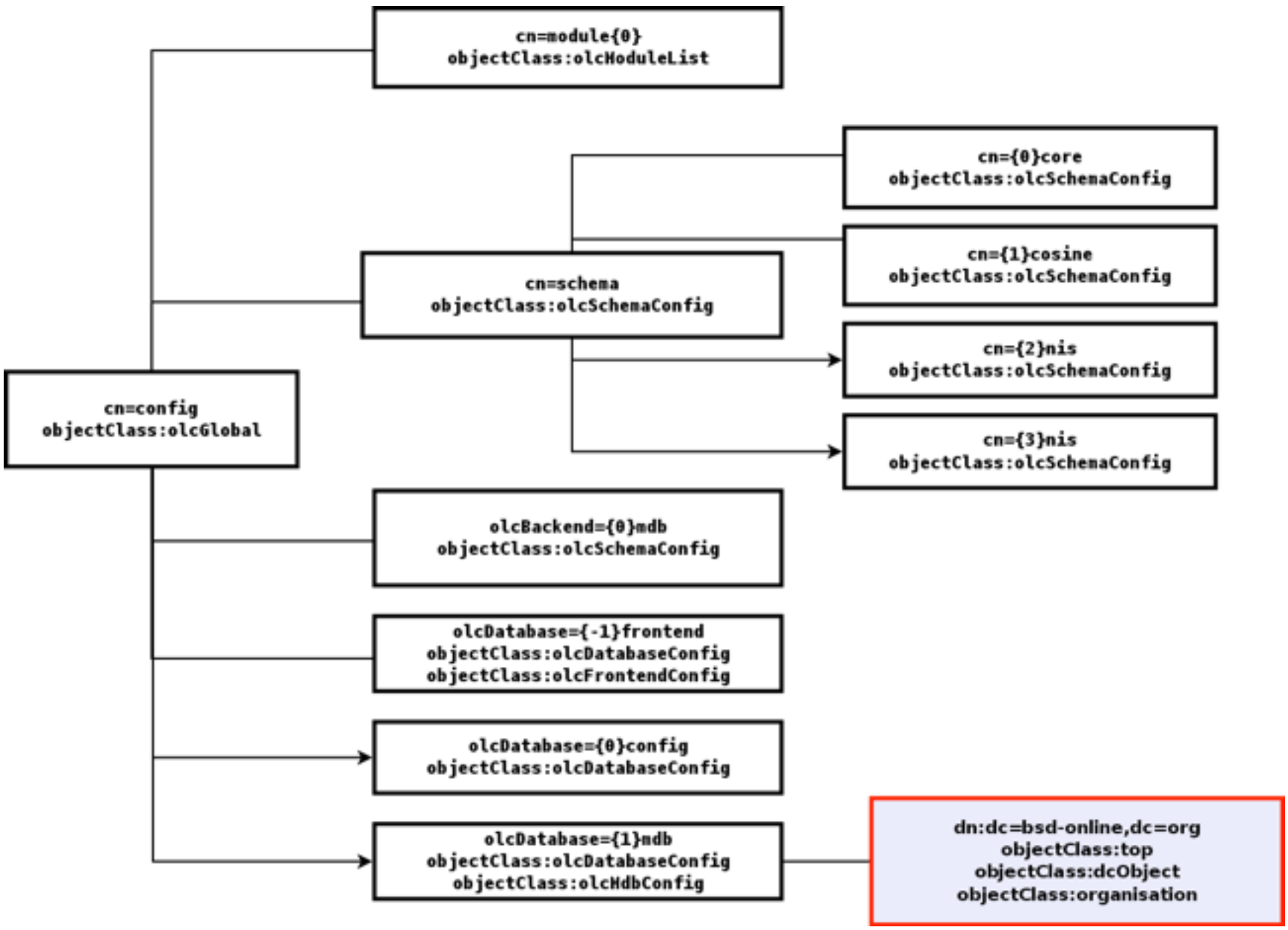


Figure 3: OpenSSL portsmaster configuration dialog

Next, we will generate a new SSL key and prepare a certificate signing request (CSR) file to be sent to a Certification Authority (CA) of your choice for signature:

```
root@laertes~# cd /usr/local/etc

root@laertes~# openssl req -sha512 -out
ldap.example.com.csr -new -newkey rsa:4096 -nodes
-keyout ldap.example.com.key
```

Afterward, we will also need to generate the required Diffie-Helmann (DH) parameters file:

```
root@laertes~# openssl dhparam -out
/usr/local/etc/dhparam.pem 4096
```

Once all of the required packages have been successfully installed, it is time to start with the preliminary analysis of LDAP databases architecture. Before starting our OpenLDAP server, edit /etc/rc.conf file and add the following entries to enable OpenLDAP on our FreeBSD system:

```
slapd_enable="YES"

slapd_flags='-h
"ldapi://%2fvar%2frun%2fopenldap%2fldapi/
ldap://0.0.0.0/"'

slapd_sockets="/var/run/openldap/ldapi"

slapd_cn_config="YES"
```

Please note that by default, the /usr/local/etc/rc.d/slapd script starts slapd(8) using only the static configuration file slapd.conf instead of our OLC-based configuration at /usr/local/etc/openldap/slapd.d/ directory. Due to this, you must take care not to forget to add the corresponding entry for slapd_cn_config to /etc/rc.conf.

If you want to use LDAP/S, modify the slapd_flags line above by adding a ldaps:/// URI as shown below:

```
slapd_flags='-h
"ldapi://%2fvar%2frun%2fopenldap%2fldapi/ ldap:///
ldaps:///"'
```

The OpenLDAP server daemon slapd(8) defaults to looking for a text configuration file named slapd.conf placed in /usr/local/etc/openldap/. This file also needs to be updated to select whichever database backend you would like to use for your data, and to make use of the `mdb` backend as we have, find and uncomment the following entry:

```
moduleload    back_mdb
```

Now, to make use of SASL authentication for LDAP, you must generate a root password by using the slappasswd(1) command:

```
root@laertes:/usr/ports # slappasswd -h "{SSHA}"

New password:

Re-enter new password:

{SSHA}bNTVGLTAytavPx55XTSE2dEs2j10An18
```

This password shall be included at the end of slapd.conf file:

```
root@laertes:# echo “rootpw
{SSHA}bNTVGLTAytavPx55XTSE2dEs2j10An18” >>
/usr/local/etc/openldap/slapd.conf
```

and defining the BaseDN and the RootDN to be used later on to deploy and administer LDAP directories:

```
#######################################################

# MDB database definitions

#######################################################

database        mdb

maxsize         1073741824

suffix          "dc=cae-hpc,dc=org"

rootdn          "cn=admin,dc=cae-hpc,dc=org"

# Cleartext passwords, especially for the rootdn,
should

# be avoid.  See slappasswd(8) and slapd.conf(5)
for details.

# Use of strong authentication encouraged.

rootpw          secret

# The database directory MUST exist prior to
running slapd AND

# should only be accessible by the slapd and slap
tools.

# Mode 700 recommended.

directory       /var/db/openldap-data

# Indices to maintain

index   objectClass     eq

olcDbIndex: uidNumber eq

olcDbIndex: uniqueMember eq

olcDbIndex: gidNumber eq

olcDbIndex: cn eq

olcDbIndex: memberUid eq
```

Also use the script to start the slapd(8) daemon:

```
root@laertes:~ # /usr/local/etc/rc.d/slapd start

Starting slapd.
```

Now, the OpenLDAP server becomes active and ready to be populated with our data. However, before starting with it, let us take some time to examine LDAP taxonomy to introduce the new approach for configuring LDAP servers: the OpenLDAP Online Configuration (OCL).

LDAP Configuration

A more complex solution to handle NIS+ based upon LDAP servers requires modifying the slapd.conf file as follows:

```
include /usr/local/etc/openldap/schema/core.schema

include
/usr/local/etc/openldap/schema/cosine.schema

include /usr/local/etc/openldap/schema/corba.schema

include
/usr/local/etc/openldap/schema/inetorgperson.schema

include /usr/local/etc/openldap/schema/nis.schema

include
/usr/local/etc/openldap/schema/collective.schema

include
/usr/local/etc/openldap/schema/openldap.schema
```

```
include
/usr/local/etc/openldap/schema/duaconf.schema

include
/usr/local/etc/openldap/schema/dyngroup.schema

include /usr/local/etc/openldap/schema/misc.schema

include /usr/local/etc/openldap/schema/pmi.schema

include
/usr/local/etc/openldap/schema/ppolicy.schema

pidfile /var/run/openldap/slapd.pid

argsfile /var/run/openldap/slapd.args

logfile /var/log/slapd.log

loglevel 256

modulepath /usr/local/libexec/openldap

moduleload back_mdb

disallow bind_anon

require authc

database mdb

suffix "dc=example,dc=org"

rootdn "cn=admin,dc=example,dc=org"

directory /var/db/openldap-data

maxsize 1073741824



access to attrs=userPassword

        by self write

        by anonymous auth

        by dn.base="cn=admin,dc=example,dc=org"
write

        by * none

access to *

        by self write

        by dn.base="cn=admin,dc=example,dc=org"
write

        by * read

# Indices to maintain

index objectClass  eq

index uid          eq

index uidNumber    eq

index uniqueMember eq

index gidNumber    eq

index cn           eq

index memberUid    eq

rootpw {SSHA}A6ia1SPQlY4J5qWBUkPg1qqiwZHrL0mb

overlay         memberof

memberof-dangling    drop

memberof-refint      TRUE
```

Traditionally, text files have been the way of setting up configuration for server daemons in Unix world. However, OpenLDAP 2.4 introduces a new way to perform this configuration using Online Configuration. Online Configuration uses an existing LDAP database to store these settings. The instrument used are special text files named LDAP Interchange Format (LDIF) files, and this new procedure of configuring and defining everything in OpenLDAP servers is known as OpenLDAP Online Configuration (OCL) method.

OpenLDAP Online Configuration (OCL)

Conventionally, OpenLDAP was configured using text files in a static way. In this case, slapd.conf file was used by default. Beside this method of configuring, OpenLDAP 2.3 and later releases also support a new dynamic and online approach of configuring LDAP known as On-Line Configuration (OLC). OLC will be used in this article.

OCL represents OpenLDAP server configuration as a tree (DIT) whose rootDN is the entity named **cn=config**. A more detailed picture of LDAPv3 organisation using Online Configuration (OLC) is depicted in Illustration 4.

Figure 4: OpenLDAP cn=config DIT Hierarchy

LDAP servers store information in hierarchical structures named Directory Information Trees (DIT). Additionally, in contrast to the former text-based configuration approach, OpenLDAP v2.4 incorporates OpenLDAP Online Configuration (OCL) which is slightly different from the former method.

All DITs are held by a super-structure named Root DSE. DSE stands for DSA Specific Entry and it acts as a management or control entity.

Recall the `slapd.conf` file we wrote in the previous section, its alternative OCL formulation is as shown below:

a) Part I. Global **cn=config** configuration

```
dn: cn=config

objectClass: olcGlobal

cn: config

#

#

# Define global ACLs to disable default read
access.

#

olcArgsFile: /var/db/run/slapd.args

olcPidFile: /var/db/run/slapd.pid

#

# Log level

#

olcLogLevel: -1

#

# Do not enable referrals until AFTER you have a
working directory

# service AND an understanding of referrals.

#olcReferral:   ldap://root.openldap.org

#

# Sample security restrictions

#       Require integrity protection (prevent
hijacking)

#       Require 112-bit (3DES or better) encryption
for updates

#       Require 64-bit encryption for simple bind

#olcSecurity: ssf=1 update_ssf=112 simple_bind=64
```

b) Part II. Dynamic Modules load

LMDB backend does not use caching. Moreover, it does not have special tuning needs to achieve a good performance, apart from index rebuilding. Therefore, LMDB is the recommended primary backend to replace the Berkeley DB backend, and, we shall select it for our OpenLDAP server configuration.

```
#

# Load dynamic backend modules:

#

dn: cn=module,cn=config

objectClass: olcModuleList

cn: module

olcModulepath:  /usr/local/libexec/openldap

#olcModuleload: back_bdb.la

#olcModuleload: back_hdb.la

#olcModuleload: back_ldap.la

#olcModuleload: back_passwd.la

#olcModuleload: back_shell.la
```

```
olcModuleload:  back_mdb.la

c) Part III. Schema load

dn: cn=schema,cn=config

objectClass: olcSchemaConfig

cn: schema

include:
file:///usr/local/etc/openldap/schema/core.ldif

include:
file:///usr/local/etc/openldap/schema/cosine.ldif

include:
file:///usr/local/etc/openldap/schema/inetorgperson
.ldif

d) Part IV. Frontend Database

dn: olcDatabase=frontend,cn=config

objectClass: olcDatabaseConfig

objectClass: olcFrontendConfig

olcDatabase: frontend

#

# Sample global access control policy:

#       Root DSE: allow anyone to read it

#       Subschema (sub)entry DSE: allow anyone to
read it

#       Other DSEs:

#               Allow self write access

#               Allow authenticated users read
access

#               Allow anonymous users to
authenticate

#

#olcAccess: to * by * read

olcAccess: to dn.base="" by * read

olcAccess: to dn.base="cn=Subschema" by * read

olcAccess: to * by self write by users read by
anonymous auth

e) Part V. MDB Database Backend

dn: olcDatabase=mdb,cn=config

objectClass: olcDatabaseConfig

objectClass: olcMdbConfig

olcDatabase: mdb

olcSuffix: dc=bsd-online,dc=org

olcRootDN: cn=admin,dc=bsd-online,dc=org

# Cleartext passwords, especially for the rootdn,
should

# be avoided.  See slappasswd(8) and
slapd-config(5) for details.

# Use of strong authentication encouraged.

olcRootPW: {SSHA}jCgbLiQs8v9kYwKpLAI6oiHPI8ZZwzca

# The database directory MUST exist prior to
running slapd AND

# should only be accessible by the slapd and slap
tools.

# Mode 700 recommended.

olcDbDirectory: /var/db/openldap-data

# Indices to maintain

olcDbIndex: objectClass eq

olcDbIndex: default pres,eq

olcDbIndex: uid

olcDbIndex: cn,sn pres,eq,sub
```

The five parts of the listings above may be joined in a single LDIF file paying attention to separate all DN using empty lines.

Eventually, to perform the initial load of slapd(8) daemon, just create a new directory to store its databases and start loading the previous LDIF file:

```
root@laertes:~ # mkdir
/usr/local/etc/openldap/slapd.d/

root@laertes:~ # slapadd -F
/usr/local/etc/openldap/slapd.d/ -n 0 -l slapd.ldif
```

As a result, the directory slapd.d is populated with a subdirectories tree starting with cn=config/ and its associated LDIF file:

```
root@laertes:/usr/local/etc/openldap/slapd.d # ls
-R

cn=config   cn=config.ldif
```

```
./cn=config:

cn=module{0}.ldif cn=schema.ldif
olcDatabase={0}config.ldif

cn=schema         olcDatabase={-1}frontend.ldif
olcDatabase={1}mdb.ldif

./cn=config/cn=schema:

cn={0}core.ldif         cn={1}cosine.ldif
cn={2}inetorgperson.ldif
```

In the event an error occurs, it is possible to clean up all OpenLDAP configuration and start from scratch by executing the commands below:

```
# cd /usr/local/etc/openldap/

# rm -fr slapd.d/*

# ../rc.d/slapd stop

# rm -fr /var/db/openldap-data/*

.# ./rc.d/slapd start
```

However, it is possible to convert a static `slapd.conf` file to the alternative OCL configuration directories tree using the `slaptest`(8) tool:

```
root@laertes:/usr/local/etc/openldap# slaptest -f
slapd.conf -F slap.conf.d/
```

For a more-in-depth view on slapd(8) configuration using OCL, check the slapd-config(5) manual page for an accurate description of the existing backends.

Navigating across LDAP Server Hierarchy. Search Patterns

According to RFC 1558, there are three types of search scopes set up in the tree hierarchy defined in every LDAP server that are used by ldapsearch(1) command:

- Base
- One-Level
- Sub-Tree

Your selection will depend on whether you are diving in the top-level tree node (Base), descent to the first immediate level of the tree hierarchy (One-Level) or diving into the whole tree hierarchy as it will be seen in the following examples:

a) Root DSE Entry

```
dn:

structuralObjectClass: OpenLDAProotDSE

configContext: cn=config

namingContexts: dc=bsd-online,dc=org

supportedControl: 1.3.6.1.4.1.4203.1.9.1.1

supportedControl: 2.16.840.1.113730.3.4.18

supportedControl: 2.16.840.1.113730.3.4.2

supportedControl: 1.3.6.1.4.1.4203.1.10.1

supportedControl: 1.3.6.1.1.22

supportedControl: 1.2.840.113556.1.4.319

supportedControl: 1.2.826.0.1.3344810.2.3

supportedControl: 1.3.6.1.1.13.2

supportedControl: 1.3.6.1.1.13.1

supportedControl: 1.3.6.1.1.12

supportedExtension: 1.3.6.1.4.1.4203.1.11.1

supportedExtension: 1.3.6.1.4.1.4203.1.11.3

supportedExtension: 1.3.6.1.1.8

supportedFeatures: 1.3.6.1.1.14

supportedFeatures: 1.3.6.1.4.1.4203.1.5.1

supportedFeatures: 1.3.6.1.4.1.4203.1.5.2

supportedFeatures: 1.3.6.1.4.1.4203.1.5.3

supportedFeatures: 1.3.6.1.4.1.4203.1.5.4

supportedFeatures: 1.3.6.1.4.1.4203.1.5.5

supportedLDAPVersion: 3

supportedSASLMechanisms: SCRAM-SHA-1

supportedSASLMechanisms: GSSAPI

supportedSASLMechanisms: GSS-SPNEGO

supportedSASLMechanisms: DIGEST-MD5

supportedSASLMechanisms: CRAM-MD5

supportedSASLMechanisms: NTLM

entryDN:

subschemaSubentry: cn=Subschema
```

b) Querying DITs managed by LDAP

The base entry of each DIT is available through the namingContexts attribute:

```
root@laertes:~ # ldapsearch -H ldap:// -x -s base
-b "" -LLL "namingContexts"

dn:

namingContexts: dc=bsd-online,dc=org
```

c) Querying DITs used for LDAP Configuration

```
root@laertes:~ # ldapsearch -H ldap:// -x -s base
-b "" -LLL "ConfigContext"

dn:

configContext: cn=config
```

d) Accessing Configuration DITs

To see all the contents of the main configuration DIT and schemas loaded, run the ldapsearch command as shown below:

```
root@laertes:~ # ldapsearch -Y EXTERNAL -H ldap:///
-b cn=config

dn: cn=config

objectClass: olcGlobal

cn: config

olcArgsFile: /var/run/openldap/slapd.args

olcPidFile: /var/run/openldap/slapd.pid

olcTLSCACertificatePath: /etc/openldap/certs

olcTLSCertificateFile: /etc/openldap/certs/cert.pem

olcTLSCertificateKeyFile:
/etc/openldap/certs/priv.pem

olcLogLevel: -1

dn: cn=schema,cn=config

objectClass: olcSchemaConfig

cn: schema

olcObjectIdentifier: OLcfg 1.3.6.1.4.1.4203.1.12.2

olcObjectIdentifier: OLcfgAt OLcfg:3

olcObjectIdentifier: OLcfgGlAt OLcfgAt:0

olcObjectIdentifier: OLcfgBkAt OLcfgAt:1

olcObjectIdentifier: OLcfgDbAt OLcfgAt:2

olcObjectIdentifier: OLcfgOvAt OLcfgAt:3

olcObjectIdentifier: OLcfgCtAt OLcfgAt:4

olcObjectIdentifier: OLcfgOc OLcfg:4

olcObjectIdentifier: OLcfgGlOc OLcfgOc:0

olcObjectIdentifier: OLcfgBkOc OLcfgOc:1

olcObjectIdentifier: OLcfgDbOc OLcfgOc:2

olcObjectIdentifier: OLcfgOvOc OLcfgOc:3

olcObjectIdentifier: OLcfgCtOc OLcfgOc:4

olcObjectIdentifier: OMsyn
1.3.6.1.4.1.1466.115.121.1

.............................

objectClass: olcDatabaseConfig

objectClass: olcFrontendConfig

olcDatabase: {-1}frontend

dn: olcDatabase={0}config,cn=config

objectClass: olcDatabaseConfig

olcDatabase: {0}config

olcAccess: {0}to * by
dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=ext
ernal
 ,cn=auth" manage by * none

dn: olcDatabase={1}monitor,cn=config

objectClass: olcDatabaseConfig

olcDatabase: {1}monitor

olcAccess: {0}to * by
dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=ext
ernal
 ,cn=auth" read by
dn.base="cn=admin,dc=bsd-online,dc=org" read by *
none

dn: olcDatabase={2}hdb,cn=config

objectClass: olcDatabaseConfig

objectClass: olcHdbConfig

olcDatabase: {2}hdb

olcDbDirectory: /var/lib/ldap

olcDbIndex: objectClass eq,pres

olcDbIndex: ou,cn,mail,surname,givenname
eq,pres,sub

olcSuffix: dc=bsd-online,dc=org
```

```
olcRootDN: cn=admin,dc=bsd-online,dc=org

olcRootPW: {SSHA}wd1zFsTyEDMtFqvzPahzSzVU0bOKicIN
```

A short look at the DN managed by DIT Configuration Entry may be displayed here:

```
root@laertes:~# ldapsearch -H ldap:// -Y EXTERNAL
-b "cn=config" -LLL -Q dn

dn: cn=config

dn: cn=module{0},cn=config

dn: cn=schema,cn=config

dn: cn={0}core,cn=schema,cn=config

dn: cn={1}cosine,cn=schema,cn=config

dn: cn={2}nis,cn=schema,cn=config

dn: cn={3}inetorgperson,cn=schema,cn=config

dn: olcBackend={0}mdb,cn=config

dn: olcDatabase={-1}frontend,cn=config

dn: olcDatabase={0}config,cn=config

dn: olcDatabase={1}mdb,cn=config
```

And the most important thing, where the slapd(8) server information is stored:

```
root@laertes2:~# ldapsearch -H ldapi:// -Y EXTERNAL
-b "cn=config" -LLL -Q -s base

dn: cn=config

objectClass: olcGlobal

cn: config

olcArgsFile: /var/run/openldap/slapd.args

olcPidFile: /var/run/openldap/slapd.pid

olcTLSCACertificatePath: /etc/openldap/certs

olcTLSCertificateFile: /etc/openldap/certs/cert.pem

olcTLSCertificateKeyFile:
/etc/openldap/certs/priv.pem

olcLogLevel: -1
```

For advanced users, to verify the current status of LDAP SSL/TLS server, GNU TLS provides an easy way to check if everything is all right:

```
root@laertes:~# gnutls-cli-debug -p 389 localhost

GnuTLS debug client 3.3.24

Checking localhost:636

unknown protocol ldaps

                                  for SSL 3.0 (RFC6101)
support... yes

                               whether we need to disable
TLS 1.2... no

                               whether we need to disable
TLS 1.1... no

                               whether we need to disable
TLS 1.0... no

                               whether %NO_EXTENSIONS is
required... no

                                 whether %COMPAT is
required... no

                                  for TLS 1.0 (RFC2246)
support... yes

                                  for TLS 1.1 (RFC4346)
support... yes

                                  for TLS 1.2 (RFC5246)
support... yes

                                  for certificate
chain order... sorted

                        for safe renegotiation (RFC5746)
support... yes

                           for Safe renegotiation support
(SCSV)... yes

                                 for heartbeat (RFC6520)
support... no

                             for version rollback bug in
RSA PMS... dunno

                        for version rollback bug in
Client Hello... no

                  whether the server ignores the RSA PMS
version... yes

                  whether small records (512 bytes) are
accepted... yes

              whether cipher suites not in SSL 3.0 spec are
accepted... yes

whether a bogus TLS record version in the client
hello is accepted... yes

               whether the server understands TLS closure
alerts... partially
```

```
            whether the server supports session
resumption... yes

                      for anonymous authentication
support... no

                      for ephemeral Diffie-Hellman
support... yes

                    for ephemeral EC Diffie-Hellman
support... yes

                     ephemeral EC Diffie-Hellman
group info... SECP256R1

                   for AES-128-GCM cipher (RFC5288)
support... yes

                   for AES-128-CBC cipher (RFC3268)
support... yes

              for CAMELLIA-128-GCM cipher (RFC6367)
support... no

              for CAMELLIA-128-CBC cipher (RFC5932)
support... no

                      for 3DES-CBC cipher (RFC2246)
support... yes

                   for ARCFOUR 128 cipher (RFC2246)
support... yes

                                        for MD5 MAC
support... yes

                                      for SHA1 MAC
support... yes

                                    for SHA256 MAC
support... yes

                                for ZLIB compression
support... no

                      for max record size (RFC6066)
support... no

                 for OCSP status response (RFC6066)
support... no

               for OpenPGP authentication (RFC6091)
support... no
```

Alternatively, from another remote computer with network access to our LDAP server:

```
root@laertes2:/etc/default# nmap -Pn -p T:636
--script ssl-enum-ciphers localhost

Starting Nmap 6.40 ( http://nmap.org ) at
2017-10-03 12:12 CEST

Nmap scan report for localhost (127.0.0.1)
Host is up (690s latency).

Other addresses for localhost (not scanned):
127.0.0.1

PORT    STATE SERVICE

636/tcp open  ldapssl

| ssl-enum-ciphers:

|   TLSv1.2:

|     ciphers:

|       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA - strong

|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong

|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 -
strong

|       TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 -
strong

|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong

|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 -
strong

|       TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 -
strong
```

At this point, the configuration of our OpenLDAP server running on FreeBSD is complete, and it is possible to perform some queries to check the accuracy of its contents

Conclusions and Remarks

While operating an OpenLDAP server can seem tricky at first, getting to know the configuration DIT and how to find meta-data within the system can help you hit the ground running. Modifying the **cn=config** DIT with LDIF files can immediately affect the running system. Likewise, configuring the system via a DIT allows you to potentially set up remote administration using only LDAP tools. This means that you can separate LDAP administration from server administration. Directory Services are a common mean to provide centralised management for an heterogeneous environment in which coexist different platform architectures and operating systems, which makes this subject specially relevant for corporate applications. Furthernore, most of the activities described in this part may be applied to other Unix-like systems in a straightforward way.

For this reason, the second part of this serie will focus in one of the most typical application of LDAP which is the embedding of NIS+ tables onto a directory structure using FreeBSD.

Acronyms and Abbreviations

DSA Directory Specific Agent
DIT Directory Information Tree
DSE DSA Specific Entry
OCL OpenLDAP Online Configuration
DN Distinguished Name
RDN Relative Distinguished Name
LDIF LDAP Data Interchange Format
LDAP Lightweight Directory Access Protocol
NSS Name Service Switch
PAM Pluggable Authentication Modules
SSL Secure Sockets Layer
TLS Transport Layer Security
SASL Simple Authentication and Security Layer
OID Object Identifier
MDB Memory-Mapped Database
LMDB Lighting Memory-Mapped Databases
YP Yellow Pages

Meet the Author

José B. Alós has developed an important part of his professional career since 1999 as an EDS employee, as a UNIX System Administrator, mainly focused on SunOS/Solaris, BSD and GNU/Linux and High-Availability solutions for industry, communications services and banking.

In 2007 he joined EADS Defense and Security, as the person responsible for providing support for end-users in aircraft engineering departments for long-term projects. These days his professional career has moved to High-Performance Computing and Simulation area within Airbus Group.

He was also Assistant Professor in the Universidad de Zaragoza (Spain), and his academic background includes a PhD in Nuclear Engineering and three MsC in Electrical and Mechanical Engineering, Theoretical Physics and Applied Mathematics.

References and Bibliography

http://www.nlc-bnc.ca/publications/1/p1-244-e.html	Directories and X.500: An Introduction
Timothy A. Howes, Gordon S. Good, Mark Smith; Macmillan Publishing, USA	Understanding and Deploying LDAP Directory Services
https://tools.ietf.org/search/rfc1558	RFC1558. A String Representation of LDAP Search FIlters
https://tools.ietf.org/search/rfc2222	RFC 2222. Simple Authentication and Security Layer (SASL)
https://tools.ietf.org/search/rfc6101	RFC6101. The Secure Sockets Layer (SSL) Protocol Version 3.0
http://www.openldap.org/	OpenLDAP Home Page
http://www.penldap.org/docs/admin24/	OpenLDAP 2.4 Administration's Guide
http://phpldapadmin.sourceforge.net	PhpLDAPadmin Home Page
https://www.freebsd.org/releases/11.1R/announce.html	FreeBSD 11.1 Release Home Page
https://www.digitalocean.com/community/tutorials/how-to-install-and-manage-ports-on-freebsd-10-1	Howto Install and Manage Ports of FreeBSD 10.1

Fluentd For Centralized Logging II: Fluentd-UI and Suricata IDS

In previous issue of BSD Magazine (Vol 11 No 06), I explained how to install Fluentd from source and from ports. I gave simple example how to configure it and in this article, I'll explain in detail how to collect our logs in the server then forward and process these logs so we can have a better and meaningful information. I'm collecting logs from Suricata (https://suricata-ids.org), which is a signature-based Intrusion Detection System (IDS) like Snort. If you'd like to know more about Suricata, check out its web site.

For this experiment, I'm using FreeBSD 10.4-RELEASE with 4 GB memory. First prepare the Fluentd installation and next is install Suricata and oinkmaster with ports as follows. Oinkmaster is a tool for updating rules and signatures for Suricata.

```
$ cd /usr/ports/security/suricata
$ sudo make install clean
```

```
$ cd
/usr/ports/security/oinkmaster
$ sudo make install clean
```

Select OK or you can tick any configuration you want. But for this experiment, just use the default configuration. We can modify our configuration later after we finish with installation and configuration. I'm using Suricata version 4.0.0 from ports. Your Suricata and oinkmaster configuration is located in /usr/local/etc/suricata and /usr/local/etc/oinkmaster, respectively. You need oinkmaster to update the rules from Suricata. We need to configure oinkmaster before we can start Suricata.

```
$ cp
/usr/local/etc/oinkmaster.conf.sample
/usr/local/etc/oinkmaster.conf
```

Open file /usr/local/etc/oinkmaster.conf and add this configuration.

```
url =
http://rules.emergingthreats.net/open/s
uricata/emerging.rules.tar.gz
```

After we update the configuration from oinkmaster.conf, we can check and download new threats signatures. Use this command as follows.

```
$ sudo oinkmaster -C
/usr/local/etc/oinkmaster.conf
/usr/local/etc/suricata/rules
```

Next, enable the Suricata on boot by adding this configuration in /etc/rc.conf.

```
suricata_enable="YES"
firewall_enable="YES"
natd_enable="YES"
```

We can then issue the command service suricata start to start the daemon. Notice that we should see information like this.

```
Starting suricata.
    19/11/2017 -- 17:02:58 - <Notice>
- This is Suricata version 4.0.0
RELEASE
```

Double check with ps ax command and the logs in /var/log/suricata/suricata.log. Next step is to configure the existing Fluentd (for installation you can check in our previous article). Fluentd comes with a friendly user interface called fluentd-ui to connect between fluentd, source and output. We need to install it first using this command.

```
$ sudo gem install -V fluentd-ui
```

After installation of fluentd-ui finishes, start the service. Default address and port for fluentd-ui are 0.0.0.0 and 9292. The user interface is web-based so we can access it anywhere. Default login user is "admin" with password "changeme" (without double quotes). You have to change the default password. After login we must define the PID file, log file and config file for fluentd. Remember we have to put the same location configuration file with existing fluentd installation.

```
    PID file :
/var/log/fluentd/fluentd-ui/fluent.pid
    Log file :
/var/log/fluentd/fluentd-ui/fluent.log
    Config file :
/usr/local/etc/fluentd/fluent.conf
```

After configuring the pid, log, and conf files, we can proceed to the dashboard. Using this interface we can manage the fluentd service (start, stop, restart). Press the "Add Source and Output" link on the left dashboard. We can see the workflow from fluentd and current settings from default configuration. In the Source Group we have File, Syslog Protocol, Monitoring Agent, HTTP, and Forwarding (receiving from another fluentd). In the Output Group we have stdout (log), Treasure Data, Amazon S3, MongoDB, ElasticSearch, and Forwarding. We need to configure two types of files. First is the source, from which logs fluentd will read. Second is the output, should we forward our output file to another application or just put it in the database.

From this experiment, I'm using File (in_tail) as source to read Suricata's logs. For the output, I'm using Elasticsearch. So we have to install Elasticsearch plugin in the server. We can use fluentd-ui plugin installation to install the Elasticsearch plugin.

Fluentd is not used for analyzing the logs so we do the analyzing process with another analytics engine. We have Suricata running and we can see the logs from /var/log/suricata. Back at the dashboard in fluentd-ui, choose "Add Source and Output" and choose "File" from the Source group. Choose the file path information the same with Suricata logs. Because we will record all logs from Suricata with syslog format and json format, so we choose suricata.log (/var/log/suricata/suricata.log) using syslog format and events from suricata with json format. Scroll down the fluentd-ui interface and we see the contents from suricata.log like this.

```
20/11/2017 -- 03:16:38 - <Warning> -
[ERRCODE: SC_WARN_IPFW_UNBIND(86)] -
Unable to disable ipfw socket: Socket
is not connected

20/11/2017 -- 03:17:06 - <Notice> -
This is Suricata version 4.0.0 RELEASE

20/11/2017 -- 03:17:10 - <Notice> - all
3 packet processing threads, 4
management threads initialized, engine
started.
```

Press next button to select file format. There are several formats, incuding syslog, nginx, json, and csv. We pick syslog and for **time_format** we have to match with the log from suricata.log. Notice that in suricata.log we see the date and time as follows.

```
20/11/2017 -- 03:17:06 - <Notice> -
This is Suricata version 4.0.0 RELEASE
```

Next is **tag** we can just put “var.*” refers to which directory suricata’s log located. Press Next button once again and we have confirmation page. Press “Update & Restart” button to finish the configuration and restart fluentd. This is from my configuration.

```
<source>

  type tail

  path /log/suricata/suricata.log

  tag var.*

  format syslog

  time_format %d/%m/%%Y -- %H:%M:%S

  pos_file /tmp/fluentd--1511123793.pos

</source>
```

This input source only logs suricata service. You have to create another input for Suricata events (eve.json). Events from Suricata log is recorded using json. This is why for this input source we use json as a format. As for **pos_file** just leave it as default. The pos_file tag acting as a record position from the log file which is suricata.log. Press the **Advanced Settings** and tick “Read from head” which means fluentd will read the file log from the first line of the log.

```
<source>

  type tail

  path /log/suricata/eve.json

  tag var.*

  format json

  time_key timestamp

  read_from_head true

  pos_file /tmp/fluentd--1511132923.pos

</source>
```

Now we configure our output plugin. Go to dashboard again and choose in Output group “Elasticsearch”. Assuming you have installed your own Elasticsearch on a different server, you can change host or any label with localhost below to your own server name or IP address. Fill the form and follow this configuration as follows.

```
<match **>

  type elasticsearch

  host localhost

  port 9200

  index_name via_fluentd

  type_name via_fluentd

  logstash_format false

  utc_index true

  hosts
http://elastic:changeme@localhost:9200

  include_tag_key false

</match>
```

Before your fluentd server can connect to your Elasticsearch server, you have to create your own index on your Elasticsearch server. This is how to create an index called “via_fluentd” in Elasticsearch. This name should be the same as the one specified above in the Output configuration.

```
curl -XPUT
'localhost:9200/via_fluentd?pretty' -H
'Content-Type: application/json' -d'

{

    "settings" : {

        "index" : {

            "number_of_shards" : 3,

            "number_of_replicas" : 2

        }

    }
```

```
}

'
```

To test our configuration and make sure that fluentd can connect to Elasticsearch, stop and start the suricata service. In fluentd stdout, we should see something like this.

```
2017-11-20 07:43:01 +0900 [info]: #0
fluentd worker is now running worker=0

2017-11-20 07:44:02 +0900 [info]: #0
Connection opened to Elasticsearch
cluster => {:host=>"localhost",
:port=>9200, :scheme=>"http",
:user=>"elastic",
:password=>"obfuscated"}
```

Now check your Elasticsearch server and send the command below.

```
andrey@nada:~$ curl -XGET
'elastic:changeme@nada.clouds.web.id:92
00/_cat/count/via_fluentd?v&pretty'
```

Your Elasticsearch server should respond with something like this.

```
epoch      timestamp count

1511099729 13:55:29  4
```

This count is a prove that our fluentd successfully sent the log into Elasticsearch server.

You can try browse your Elasticsearch server using your browser using this address http://localhost:9200/via_fluentd

It should work with json format output with information taken from suricata and forwarded by fluentd.

Conclusion

Fluentd is very modular and flexible. Almost all logs from known applications can be processed and forwarded. This makes fluentd very flexible and make it easier for system administrator to manage their logs system.

Meet the Author

Andrey Ferriyan is a writer, researcher and practitioner. Python and R enthusiasts. Experiences in UNIX-like servers (GNU/Linux, FreeBSD and OpenBSD). Data Scientist wannabe. Area of interests including Information Security, Machine Learning and Data Mining.

Now He is a student at Keio University under LPDP (Indonesia Endowment Fund for Education). He leads startup company in Indonesia called ATSOFT with my friends.

FreeBSD, Google Cloud, and Dual ECC/RSA Let's Encrypt Certificates

Here's how I deployed a website to FreeBSD on the Google Cloud Platform. I set up HTTPS with free Let's Encrypt (letsencrypt.org) TLS certificates for both RSA and ECC, and set up automatic renewal of the dual certificates.

None of this is difficult, but I discovered that some steps aren't openly supported or well documented. Specifically, running FreeBSD on Google's IaaS or Infrastructure as a Service cloud environment, and automatically renewing dual RSA/ECC Let's Encrypt certificates.

This article is aimed at people who are in a situation similar to mine when I started. First, I'll assume you're reasonably comfortable with FreeBSD — no need to explain why it's a great choice for OS, or how to use the pkg command and control the Apache service.

Second, I expect that you're familiar with public cloud concepts and terminology, but you don't necessarily have any experience with Google's specific offerings.

Are you still with me? Then let's get started!

Google Cloud Platform and its Free Tier

The Google Compute Engine provides high performance, and the price is certainly right! The Free Tier includes several products that are always free up to some usage limits, of course, with a low cost beyond that. For details, see: https://cloud.google.com/compute/

The Free Tier includes one VM with plenty of horsepower for a website. Their f1-micro instance gives you a single-core Intel Xeon 2.20 GHz CPU with 614 MB of RAM. It's a shared-core machine, and you get 20% of a virtual CPU all the time with bursts up to 100%. After the first one, each additional f1-micro machine costs just US$ 3.88 per month.

The f1-micro VM comes with 30 GB of constant disk storage based on locally attached solid-state drives. That's right, the storage is all SSD, mechanical disks aren't even a choice. Additional storage is US$ 0.04 per GB per month, although 30 GB was more than enough for me.

You get one static external IPv4 address. IPv6 is currently only available when you are also using load balancers, but they say general purpose IPv6 is set for release. Google's data centers have plenty of bandwidth. Ingress traffic is unlimited, and most of the first 1 GB egress traffic per month is free. Beyond the first gigabyte of outbound traffic, the pricing is complicated but quite cheap.

The first free gigabyte is to all destinations other than Australia and China other than Hong Kong. It's US$ 0.12 per GB to most of the world after the first free gigabyte. All traffic to Australia is US$ 0.19/GB, and all traffic to China other than Hong Kong is US$ 0.23/GB.

Reserve an IP Address

Follow Google's instructions to specify your geographic region. The Free Tier is only available in some regions. My server is located in Oregon. Then, follow their instructions to reserve an external IP address. Once your VM is running and associated with that address, you can go back and specify that it should be static, not changing after a reboot.

Your initial steps are done through a web interface. I have used the AWS or Amazon Web Services dashboard on a number of projects. However, within 10 minutes of my first exposure to the Google Cloud Platform dashboard, I found it *much* more intuitive and informative.

Below is the view after the VM was running and using the reserved IP address.

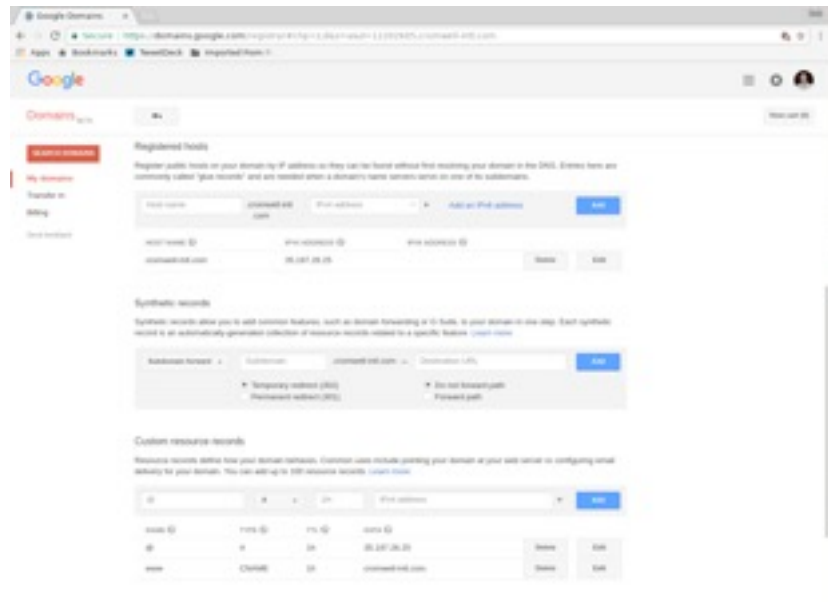
Set Up DNS

US$ 12 transfers a domain from your current registrar to Google Domains, and adds 1 year of registration. After that, it's US$ 12 per year. That's low cost considering that Google's DNS service provides great performance. Below is the Domains' dashboard where I have registered cromwell-intl.com by IP address and set up A and CNAME records. It's very easy to use.

The A record for "@", means the domain itself and it defines the IPv4 address.

The CNAME record specifies that www.cromwell-intl.com is an alias, and the canonical name is simply cromwell-intl.com.

Therefore, regardless of the user's assumption that the name has the "www." or not, it resolves to the same IP address. In a later step, I will configure Apache to redirect all requests for the "www." version to the simpler name. The search engines will see the site as a single site, not a collection duplicated across two hostnames.



Deploying the VM

There isn't a simple point-and-click method to choose a FreeBSD VM image. It appears as if FreeBSD isn't a choice! However, FreeBSD images *are* available through the freebsd-org-cloud-dev project.

FreeBSD is supported on the Google Cloud Platform because it works just fine. There isn't support in the form of *assistance,* they don't make it as simple as other operating systems. But there is an easy command-line way to deploy it.

First, install the Google Cloud SDK package on your local system. There is a google-cloud-sdk FreeBSD package, or get it for various operating systems from: https://cloud.google.com/sdk/downloads

This gives you the gcloud command-line interface set-up to run under Bash. I found gcloud *much* easier to set up and use than the corresponding AWS command-line toolkit. Once the server is deployed, you can connect with SSH, and you seldom need gcloud.

Start by using gcloud to see the list of images currently available from the FreeBSD project:

```
$ gcloud compute images list \
      --project freebsd-org-cloud-dev \
      --no-standard-images
```

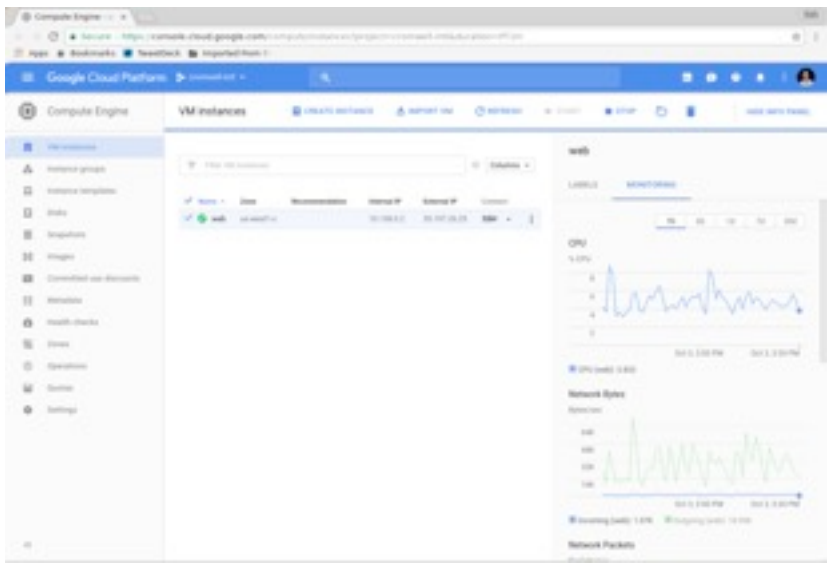
That's a lot! Let's narrow that down to the stable RELEASE versions:

```
$ gcloud compute images list \
      --project freebsd-org-cloud-dev \
      --no-standard-images | grep -i release
```

Now, deploy your FreeBSD server. Change the version as needed, and change *web* to your desired hostname. The 30 GB disk size is the maximum size for the free tier. It was much more than enough to hold my site.

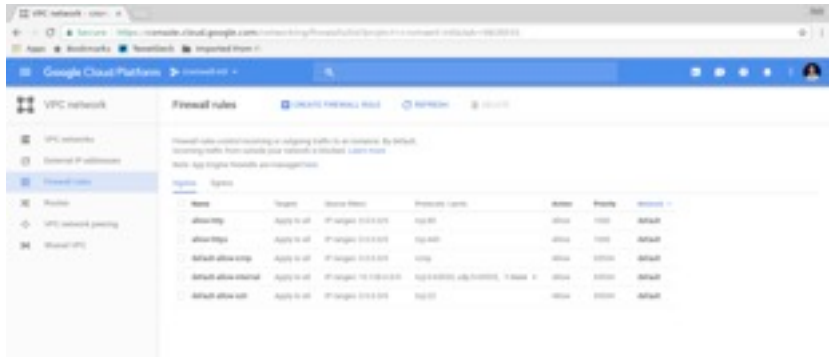
```
$ gcloud compute instances create web \
      --image-project=freebsd-org-cloud-dev \
      --image=freebsd-11-1-release-amd64 \
      --boot-disk-size=30GB \
      --boot-disk-type=pd-standard \
      --machine-type=f1-micro
```

Now you can start the VM through the web dashboard. It's running!



Set Up SSH

Verify that the virtualized firewall will pass inbound SSH. Let's go ahead and add HTTP and HTTPS, and remove the unneeded rule allowing RDP. We'll also make sure that ICMP is allowed, so we can do simple ping tests.



Follow Google's instructions to add SSH keys:

https://cloud.google.com/compute/docs/instances/adding-removing-ssh-keys

Once I get a basic .ssh/authorized_keys file in place, I do everything with ssh and scp. The web interface is just for general monitoring and, *maybe,* rebooting.

Your user can become root with sudo bash, and you can make further changes. Do not assign passwords to any user, stick with cryptographic authentication over SSH.

If you add a user to group wheel, they can become root by simply running the command su because of the contents of /etc/pam.d/su.

Networking

The Ethernet interface will be vtnet0. Your server is in a private network, a VPC or Virtual Private Cloud, something like the 10.138.0.0/24 network with just your server and a (virtual) router. The router runs NAT, mapping your server to your external static IP address.

Packages

The FreeBSD image comes with several packages added to the basic install. I found these 22 packages on the freshly deployed image:

```
bash, ca_root_nss, curl,
firstboot-freebsd-update, firstboot-growfs,
flock, gettext-runtime, google-cloud-sdk,
google-daemon, google-startup-scripts,
indexinfo, libffi, libnghttp2, panicmail,
pkesh, pkg, python, python2, python27,
readline, sudo.
```

I first installed all available updates for the existing packages. Then, I added bind-tools and lsof for troubleshooting, and vim for my personal preference.

Thereafter, I added the packages needed for Apache/PHP web service: apache24 and mod_php71 and their dependencies.

Correcting Clock Problems

By now, the system has been running long enough that I noticed the *huge* clock drift. Within a minute, the system clock drifts by several seconds. This is a known issue with FreeBSD running on Linux/KVM. Let's see what the virtualized platform provides:

```
$ dmesg | less
[... output deleted ...]
random: unblocking device.
ioapic0 <Version 1.1> irqs 0-23 on motherboard
Timecounter "TSC" frequency 1837606598 Hz
quality 1000
random: entropy device external interface
[... output deleted ...]
atrtc0: <AT realtime clock> port
0x70-0x71,0x72-0x77 irq 8 on acpi0
Event timer "RTC" frequency 32768 Hz quality 0
Timecounter "ACPI-fast" frequency 3579545 Hz
quality 900
acpi_timer0: <24-bit timer at 3.579545MHz>
port 0xb008-0xb00b on acpi0
[... output deleted ...]
attimer0: <AT timer> at port 0x40 on isa0
Timecounter "i8254" frequency 1193182 Hz
quality 0
attimer0: Can't map interrupt.
ppc0: cannot reserve I/O port range
Timecounters tick every 1.000 msec
[... output deleted ...]
```

You want to use the ACPI-fast device:

```
# sysctl kern.timecounter.hardware
kern.timecounter.hardware: TSC
# sysctl kern.timecounter.choice
kern.timecounter.choice: i8254(0)
ACPI-fast(900) TSC(1000) dummy(-1000000)
# sysctl kern.timecounter.hardware=ACPI-fast
kern.timecounter.hardware: TSC -> ACPI-fast
```

I then added a line to each of /etc/sysctl.conf and /etc/rc.conf.

```
$ grep timecounter /etc/sysctl.conf
kern.timecounter.hardware=ACPI-fast
# grep ntpd /etc/rc.conf
ntpd_enable=YES
ntpdate_enable=YES
```

I rebooted to test, and now the clock was correct and stayed spot-on.

Set Up Apache

I set-up Apache 2.4 with PHP 7.1, and got the site served out over HTTP. This is well-documented elsewhere. So let's move to the next step.

Public-Key Security

Asymmetric cryptography, also called public-key cryptography, bases its security on a *trapdoor function.* The trapdoor function is easy to compute in one direction, but difficult to compute in the opposite direction. RSA, which was developed in the late 1970s, relies on the difficulty of factoring the product of two very large prime numbers. It is easy to multiply integers, even the ones with hundreds of digits. However, it is impractically difficult to start with such a product and figure out which two large prime numbers went into it.

Then people got worried: what if someone develops a general-purpose quantum computer? *Shor's Algorithm* could quickly factor very large numbers if you run it on such a platform.

Around this time, people started using mobile devices for Internet access. However, smartphones with fast

multi-core CPUs had not yet been developed. We're talking about early BlackBerry days.

So, *Elliptic Curve Cryptography* or ECC suddenly became popular. Its trapdoor function is based on a discrete logarithm, entirely different from RSA's factoring. Analysis by NSA and NIST showed that ECC provides same security with *much* smaller keys than RSA, requiring much less computation.

So, two advantages: higher performance, and perceived resistance to sudden obsolescence when quantum computers appear. Certificate authorities began issuing dual certificates for sites: one based on ECC which newer clients would prefer for performance, and RSA as a fall-back.

Since then, cryptographers have discovered that ECC will be just as susceptible as RSA to attack by quantum computers. But ECC still has a performance advantage.

In August 2015, the NSA announced that ECC wasn't a backup for RSA when facing the threat of quantum computing cryptanalysis, to the point that government agencies and contractors considering a migration from RSA to ECC shouldn't bother. They later modified the page, and thereafter, took it down. See the archived update here.

We need **post-quantum** or **quantum-safe** asymmetric ciphers. Several families of Key post-quantum cipher algorithms are being explored: lattice-based cryptography, code-base cryptography, multivariate polynomial cryptography, and others. See the Post-Quantum Crypto conference series for details.

To get back on track, ECC has a definite performance advantage over RSA at the same security levels. We want to support both.

TLS with Dual ECC/RSA Let's Encrypt Certificates

Let's Encrypt (letsencrypt.org) is a certificate authority founded by the Electronic Frontier Foundation, the Mozilla Foundation, the University of Michigan, Akamai Technologies, and Cisco Corporation. They issue free TLS certificates that are trusted by browsers. (Advanced note: these are DV or Domain Verification certificates, not EV or Extended Verification, limiting browsers' trust in them, but the price is certainly right!)

Let's Encrypt certificates are only good for 90 days. The short certificate lifetime makes automated renewal important.

Yes, you *can* set up automated renewal of dual ECC/RSA Let's Encrypt certificates! I found that this wasn't documented very well. Google searches lead to frequently-referenced blog postings about it being impossible, or how there is an overly complex kludge when working around it. Hence the main point of this article is that it's not hard to figure out, and it's quite easy to set it up once you know the trick

Creating the RSA Certificate

ACME, the Automated Certificate Management Environment, is a protocol for interacting with the Let's Encrypt CA. You use the **certbot** program to carry out the various steps. Install the py27-certbot package to get it.

Now you're ready to make your first certificate:

```
# certbot certonly --webroot \
      -w /usr/local/www/htdocs/ \
      -d example.com -d www.example.com
```

I deliberately provided the root location of the web document, and listed the domain names. Yes, clients will be redirected from www.example.com to example.com as needed, but they must first make a secure connection server and ask for the longer name with "www.".

There is some narrative output. You are asked for an email address in case they need to send you an urgent renewal or security notice. You agree to the terms of service, then answer whether it's OK to share your email address with the EFF, and you are done.

I didn't tell it anything about the cryptography, so it generated and installed a 2048-bit RSA key pair.

What Did You Get?

The key pair, certificate, and associated files have been created and saved under /usr/local/etc/letsencrypt. Let's see what files were saved there.

```
# cd /usr/local/etc/letsencrypt
# tree -F
.
|-- accounts/
|   |-- acme-staging.api.letsencrypt.org/
|   |   `-- directory/
|   |       `--
d72ae2a5cf968487add7cbdece6e3aab/
|   |           |-- meta.json
|   |           |-- private_key.json
|   |           `-- regr.json
|   `-- acme-v01.api.letsencrypt.org/
|       `-- directory/
|           `--
5f78856fecb3b21a157f41d986716e2c/
|               |-- meta.json
|               |-- private_key.json
|               `-- regr.json
|-- archive/
|   `-- example.com/
|       |-- cert1.pem
|       |-- chain1.pem
|       |-- fullchain1.pem
|       `-- privkey1.pem
|-- csr/
|   `-- 0000_csr-certbot.pem
|-- keys/
|   `-- 0000_key-certbot.pem
|-- live/
|   `-- example.com/
|       |-- README
|       |-- cert.pem ->
../../archive/example.com/cert1.pem
|       |-- chain.pem ->
../../archive/example.com/chain1.pem
|       |-- fullchain.pem ->
../../archive/example.com/fullchain1.pem
|       `-- privkey.pem ->
../../archive/example.com/privkey1.pem
`-- renewal/
    `-- example.com.conf

14 directories, 18 files
```

Notice the directories archive, containing the key files, and live, containing links to those files. We will return to this detail in a bit.

Creating the ECC Certificate

Let’s now create an ECC private key and certificate. We need a reasonably recent version of openssl. Check what yours is capable of:

```
$ openssl ecparam -list_curves | less
```

I will use elliptic curve P-384, designated secp384r1, as it is the strongest elliptic curve included in NSA Suite B cryptography. See the U.S. NIST SP 800-57 "Recommendation for Key Management" for its definition, and the following comparison of relative strength against brute force attack:

Key Length in Bits for Approximately Equal Resistance to Brute-Force Attacks, per NIST/NSA			
Security Strength	Symmetric (3DES, AES)	Asymmetric (RSA, DSA)	Elliptic Curve
80	80	1024	160
112	112	2048	224
128	128	3072	256
192	192	7680	384
256	256	15,360	512

The first time I used certbot, I let it generate an RSA key pair. **Since I need to generate an ECC certificate-signing request, I’ll** start by generating an ECC private key:

```
$ openssl ecparam -genkey -name secp384r1 | openssl
ec -out ecc-privkey.pem
```

Before generating the CSR or Certificate Signing Request, I must slightly change the OpenSSL configuration to enable multiple names, both with and without "www.":

Edit /etc/ssl/openssl.cnf.

Find and uncomment the entry:

req_extensions = v3_req

Add a line below that:

subjectAltName = @alt_names

Add a new stanza at the end of the file:

Added

[alt_names]

DNS.1 = www.example.com

DNS.2 = example.com

I can now generate the CSR. It will ask you for a 2-letter country code, state or province, locality, and so on.

```
$ openssl req -new -sha256 -key ecc-privkey.pem
-nodes -outform pem -out ecc-csr.pem
```

Ask Let's Encrypt to generate a certificate. This time we pass it our new CSR.

```
$ certbot certonly -w /usr/local/www/htdocs \
      -d example.com -d www.example.com \
      --email bob.cromwell@comcast.net \
      --csr ecc-csr.pem --agree-tos
```

This gives us three new files in the local directory:

0000_cert.pem = The certificate itself

0000_chain.pem = The signing chain

0001_chain.pem = The full chain including our certificate

Solving the Mystery — Automatically Renewing Dual Certificates

It took some research after initial frustration to learn that certbot is *very* fussy about file names when it comes to renewal.

By default, it generates RSA keys with directory archive/*example.com* containing the actual files, and live/*example.com* containing symbolic links pointing to them. **You can rename the archive and live directories, but the files *must* have specific names.**

The "archive" directory, or whatever you end up naming it, must have files named precisely cert1.pem, chain1.pem, fullchain1.pem, and privkey1.pem.

The "live" directory, again possibly renamed, must have symbolic links with those same names minus the "1", precisely cert.pem, chain.pem, fullchain.pem, and privkey.pem.

The automated RSA installation also created a directory named renewal containing a configuration file named for the domain plus ".conf".

First, I rearranged the existing hierarchy under /usr/local/etc/letsencrypt.

Rename the existing "archive" and "live" directories rsa-archive and rsa-live.

Recreate the symbolic links in rsa-live/example.com to point to the relocated "archive" files.

Edit renewal/example.com.conf and make corresponding changes to the paths.

Rename that file rsa-example.com.conf.

Verify that renewal still works:

certbot renew --dry-run

Next, create new directories ecc-archive and ecc-live, each with a subdirectory named for the domain. Then:

Move the ECC files I just created into the ecc-archive area, **changing the names as required.**

Create the symbolic links under ecc-live.

Rename the RSA files in csr and keys, and move the corresponding ECC files into those areas.

Copy the file in renewal to ecc-*example.com*.conf and edit that new file so its contents refer to the ECC files.

The result of all this is the following, where:

yellow indicates renamed files and changed file content,

green indicates (re)created symbolic links,

blue indicates new files and directories, and

grey indicates unchanged files

```
# cd /usr/local/etc/letsencrypt
# tree -F
.
|-- accounts/
|   |-- acme-staging.api.letsencrypt.org/
|   |   `-- directory/
|   |       `-- d72ae2a5cf968487add7cbdece6e3aab/
|   |           |-- meta.json
|   |           |-- private_key.json
|   |           `-- regr.json
|   `-- acme-v01.api.letsencrypt.org/
|       `-- directory/
|           `-- 5f78856fecb3b21a157f41d986716e2c/
|               |-- meta.json
|               |-- private_key.json
|               `-- regr.json
|-- csr/
|   |-- ecc-csr.pem
|   `-- rsa-csr.pem
|-- ecc-archive/
|   `-- example.com/
|       |-- cert1.pem
|       |-- chain1.pem
|       |-- fullchain1.pem
|       `-- privkey1.pem
|-- ecc-live/
|   `-- example.com/
|       |-- cert.pem -> ../../ecc-archive/example.com/cert1.pem
|       |-- chain.pem -> ../../ecc-archive/example.com/chain1.pem
|       |-- fullchain.pem -> ../../ecc-archive/example.com/fullchain1.pem
|       `-- privkey.pem -> ../../ecc-archive/example.com/privkey1.pem
|-- keys/
|   |-- ecc-privkey.pem
|   `-- rsa-privkey.pem
|-- renewal/
|   |-- ecc-example.com.conf
|   `-- rsa-example.com.conf
|-- rsa-archive/
|   `-- example.com/
|       |-- cert1.pem
|       |-- chain1.pem
|       |-- fullchain1.pem
|       `-- privkey1.pem
`-- rsa-live/
    `-- example.com/
        |-- README
        |-- cert.pem -> ../../rsa-archive/example.com/cert1.pem
        |-- chain.pem -> ../../rsa-archive/example.com/chain1.pem
        |-- fullchain.pem -> ../../rsa-archive/example.com/fullchain1.pem
        `-- privkey.pem -> ../../rsa-archive/example.com/privkey1.pem

18 directories, 29 files
```

The automated renewal files now contain the following:

```
# cat renewal/ecc-example.com.conf
# renew_before_expiry = 30 days
version = 0.18.2
archive_dir =
/usr/local/etc/letsencrypt/ecc-archive/example.com
cert =
/usr/local/etc/letsencrypt/ecc-live/example.com/cert.pem
privkey =
/usr/local/etc/letsencrypt/ecc-live/example.com/privkey.pem
chain =
/usr/local/etc/letsencrypt/ecc-live/example.com/chain.pem
fullchain =
/usr/local/etc/letsencrypt/ecc-live/example.com/fullchain.pem

# Options used in the renewal process
[renewalparams]
authenticator = webroot
installer = None
account = 5f78856fecb3b21a157f41d986716e2c
webroot_path = /usr/local/www/htdocs,
[[webroot_map]]
www.example.com = /usr/local/www/htdocs
example.com = /usr/local/www/htdocs

# cat renewal/rsa-example.com.conf
# renew_before_expiry = 30 days
version = 0.18.2
archive_dir =
```

```
/usr/local/etc/letsencrypt/rsa-archive/example.com
cert =
/usr/local/etc/letsencrypt/rsa-live/example.com/cert.pem
privkey =
/usr/local/etc/letsencrypt/rsa-live/example.com/privkey.pem
chain =
/usr/local/etc/letsencrypt/rsa-live/example.com/chain.pem
fullchain =
/usr/local/etc/letsencrypt/rsa-live/example.com/fullchain.pem

# Options used in the renewal process
[renewalparams]
authenticator = webroot
installer = None
account = 5f78856fecb3b21a157f41d986716e2c
webroot_path = /usr/local/www/htdocs,
[[webroot_map]]
www.example.com = /usr/local/www/htdocs
example.com = /usr/local/www/htdoc
```

Now test this:

```
# certbot renew --dry-run
```

Did it work? Great!

Automated Renewal

Now, let's automate the renewal. Set up a crontab job to run certbot in renewal mode twice a day. It won't do anything until there are 30 days left. We'll do this frequently so we can spot any problems quickly. Pick random times:

```
# crontab -l
# min hr day-of-month month day-of-week command
44  4 * * * certbot renew > /root/certbot-output
2>&1
44 16 * * * certbot renew > /root/certbot-output
2>&1
# cat /root/certbot-output
Saving debug log to
/var/log/letsencrypt/letsencrypt.log
Cert not yet due for renewal
Cert not yet due for renewal


-------------------------------------------------
Processing
/usr/local/etc/letsencrypt/renewal/rsa-example.com.
conf
-------------------------------------------------


-------------------------------------------------
Processing
/usr/local/etc/letsencrypt/renewal/ecc-example.com.
conf
-------------------------------------------------


-------------------------------------------------


The following certs are not due for renewal yet:

/usr/local/etc/letsencrypt/rsa-live/example.com/fullchain.pem (skipped)

/usr/local/etc/letsencrypt/ecc-live/example.com/fullchain.pem (skipped)
No renewals were attempted.
-------------------------------------------------
```

Unless you run certbot with the --force-renewal option, it will wait until there are only 30 days left.

We can use the openssl tool to parse and display the certificates.

```
# openssl x509 -in ecc-live/example.com/cert.pem
-text -noout

# openssl x509 -in rsa-live/example.com/cert.pem
-text -noout
```

Enabling HTTPS With Those Dual Certificates

Edit the httpd.conf configuration file and add the following to the file, changing the hostname and file system paths as needed. Make sure to use the file fullchain.pem, which contains the full certificate chain, and not cert.pem which has just your site's certificate.

```
# Put these directives at the global level:
LoadModule ssl_module libexec/apache24/mod_ssl.so
Listen 443


# Put these within individual VirtualHost stanzas
# if you are hosting several sites on one server.
<VirtualHost *:443>
    ServerName example.com
    SSLEngine on
    # ECC secp384r1
    SSLCertificateFile
"/usr/local/etc/letsencrypt/ecc-live/example.com/fu
```

```
llchain.pem"
    SSLCertificateKeyFile
"/usr/local/etc/letsencrypt/ecc-live/example.com/privkey.pem"
    # RSA
    SSLCertificateFile
"/usr/local/etc/letsencrypt/rsa-live/example.com/fullchain.pem"
    SSLCertificateKeyFile
"/usr/local/etc/letsencrypt/rsa-live/example.com/privkey.pem"
</VirtualHost>
```

Restart Apache, and verify that you can connect with both HTTP and HTTPS.

Redirect to HTTPS without "www."

The goal is to accept all connections, redirecting all of these:

http://example.com/some/path/

http://www.example.com/some/path/

https://www.example.com/some/path/

to this:

https://example.com/some/path/

Add the following to your .htaccess file in the root of the web site. If the RewriteEngine line is already in the file, don't duplicate it.

```
# Remove "www." and redirect HTTP to HTTPS
RewriteEngine on
# Use a standard variable and a tagged regular expression to
# replace the URL with "https://", the host name, and the
# path minus any leading "www.":
RewriteCond %{HTTP_HOST} ^www\.(.*)$ [NC]
RewriteRule ^(.*)$ https://%1/$1 [R=301,L]
# If they asked for the non-www name but with HTTP, build a
# new HTTPS URL with the host name and the path:
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$
https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

Now test the various redirection cases.

Yes, I did it with the site-specific .htaccess file. I could have instead done it with slightly different syntax in the server-wide httpd.conf configuration file. For the single site, given its size and traffic, I didn't see a big advantage of one method over the other.

Improving the TLS Configuration

Apache has a good SSL/TLS how-to document: https://httpd.apache.org/docs/2.4/ssl/ssl_howto.html

Even more useful, Mozilla has a configuration generator: https://mozilla.github.io/server-side-tls/ssl-config-generator/

Select your server, its version, and your OpenSSL versions, and lastly, select the security profile.

Which security profile should you use? It depends...

Let's say you're setting up a server for use within your organization, and you have full control of the desktop systems and any portable laptops that could be connected from outside. In that case, I recommend the strictest "Modern" profile. All your client machines will need to be fairly current, but that should already be the case.

However, let's say that you want to be open to all clients from the public. That's my situation. It would be nice if everyone used up-to-date operating systems and browsers, but I don't want to block or even inconvenience people with outdated platforms.

I used the "Intermediate" profile as a starting point. Here is what I added towards the end of the httpd.conf configuration file, before and outside the VirtualHost stanza. Hence it will apply to *all* virtually hosted websites I eventually set up on the server. The SSLCipherSuite line is enormously long. I started with what Mozilla's "Intermediate" profile gave me, and reordered that to put 3DES or "DES-CBC3" at the end.

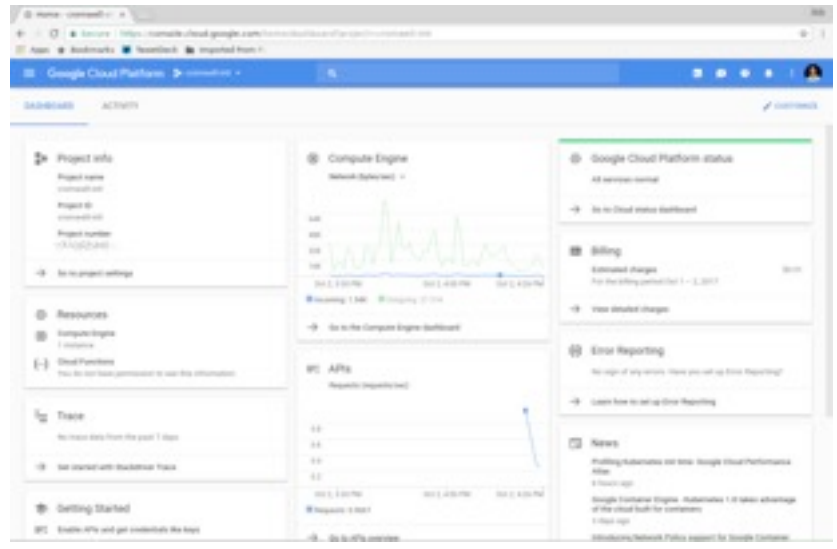
That provides 3DES as a fallback position for connections from IE 8 on XP.

```
# TLS only, no SSL
SSLProtocol all -SSLv2 -SSLv3
# Specify ciphers in a preferred order.  I
reordered what the configuration
# generator gave me, putting 3DES ("DES-CBC3") at
the end.
SSLCipherSuite
ECDHE-ECDSA-CHACHA20-POLY1305:[...much deleted, see
above]
SSLHonorCipherOrder     on
# Disable compression and session tickets
SSLCompression          off
SSLSessionTickets       off
# Enable OCSP Stapling
LoadModule socache_shmcb_module
libexec/apache24/mod_socache_shmcb.so
SSLUseStapling On
SSLStaplingCache "shmcb:logs/ssl_stapling(32768)"
# Enable session resumption (caching)
SSLSessionCache "shmcb:logs/ssl_scache"
# Insist on HSTS or HTTP Strict Transport Security
Header always set Strict-Transport-Security
"max-age=31536000; includeSubDomains; preload"
```

And with that, an A+ evaluation from the Qualys analyzer! See how your server scores at www.ssllabs.com.

Monitoring The Dashboard

You can get a quick and clear overview of recent server activity with the Google Cloud Platform dashboard. That's "Home" in the 3-line menu at the upper left in the GCP's pages. You can monitor the network traffic and CPU utilization, and keep an eye on the month's billing so far, among other features. Also, you can customize what you see here. I have network traffic and monthly billing up top and CPU utilization below the traffic graph. To me, this seems like a big improvement over the AWS dashboard, where I have to track down the pieces on various screens.



Try It Out!

I would suggest that you try FreeBSD on the Google Cloud Platform. There's zero cost for a short experiment, and I think many people will like that environment.

And More...

I have more details on my site with further Apache details, including setting up some HTTPS headers that can further enhance security. To read further, visit:

https://cromwell-intl.com/open-source/google-freebsd-tls/

Meet the Author

Bob Cromwell has been using OpenBSD since, well, not sure how long… Some time in the late 1990s. He's used Linux since you downloaded 40+ floppy images, some time around 1993-1994. Before that he had used UNIX, SunOS and forms of BSD, at Purdue since the mid 1980s. He got a BSEE at Purdue back then, worked at the university, grad school, Ph.D. in electrical and computer engineering, has done consulting since 1992. He's taught courses for Learning Tree International since the mid 1900s, and has written courses for them since the late 1990s.

HEY GOLIATH...

MEET DAVID

TRUENAS® PROVIDES MORE PERFORMANCE, FEATURES, AND CAPACITY PER-DOLLAR THAN ANY ENTERPRISE STORAGE ARRAY ON THE MARKET.

Introducing the TrueNAS X-Series: Perfectly suited for core-edge configurations and enterprise workloads such as backups, replication, and file sharing.

- ★ **Unified:** Simultaneous SAN, NAS, and object protocols to support multiple applications
- ★ **Scalable:** Up to 120 TB in 2U and 720 TB in 6U
- ★ **Safe:** High Availability ensures business continuity and avoids downtime
- ★ **Reliable:** Uses OpenZFS to keep data safe
- ★ **Trusted:** TrueNAS is the Enterprise version of FreeNAS®, the world's #1 Open Source SDS
- ★ **Enterprise:** Enterprise-class storage including unlimited instant snapshots and advanced storage optimization at a lower cost than equivalent solutions from Dell EMC, NetApp, and others

The TrueNAS X10 and TrueNAS X20 represent a new class of enterprise storage. Get the full details at iXsystems.com/TrueNAS.

iXsystems™

Mongoose Embedded Web Server on FreeBSD

You will learn …

- What is an Embedded Software
- What is a Web Server
- What is Mongoose
- How To Install Mongoose On FreeBSD
- Serving A Web Site With Mongoose
- How To Secure Mongoose Web Server

What is an Embedded Software?

An embedded software is a computer program created to control specific devices. Typically, these devices have some memory, storage, and performance limitations.

An embedded software program must be stable, clean, and fast. Memory footprint of embedded software is so critical that developers must create it with caution.

No matter it's a TV or a missile, embedded software must to perform task flawless. Embedded software needs to include all required device drivers. The device drivers are written for the specific hardware. The software is highly dependent on the CPU and specific chips chosen.

Software development requires the use of a cross-compiler which runs on a computer but produces executable code for the target device. Debugging requires the use of an in-circuit emulator, JTAG or SWD. Software developers often have access to the complete kernel (OS) source code. These limitations force developers to use C or embedded C++.

What is a Web Server?

A web server is a software system that processes requests via HTTP or any other protocols. The primary function of a web server is to manage communication between client and server, and this takes place using the Hypertext Transfer Protocol (HTTP).

Web servers are not only used for serving the internet but also as a part of a system for monitoring or administering. All we need is a browser to access these embedded applications.

There are two types of web servers:

- User-mode web server
- Kernel-mode web server

User-mode web servers are slower because the system takes time to respond to the request for allocation of resources, but are more secure. In situation where the web server is compromising, only the web server process may be dropped at crash.

A kernel-mode web server can process more queries per second or QPS.

What is Mongoose?

Mongoose is a cross-platform embedded web server which is available under GPL v2 and commercial licenses, and has a small size.

Mongoose is built on top of the Mongoose Embedded Library which can be used for the implementation of RESTful services to serve Web GUI on embedded devices. Mongoose is a cross-platform application that can be used on Windows, Macintosh OS, Linux, QNX, eCOS, Free RTOS, Android, and iOS.

With just over 130 kB source code and an executable footprint of 43 kB on FreeBSD, Mongoose is one of the smallest web servers available. Mongoose is written in C.

Mongoose is used by several companies in various industries, including software companies, equipment companies, semiconductor companies, and some Fortune 500 technology companies. In January 2017, Mongoose surpassed 2,000,000 record of downloads.

Functions of Mongoose include:

- Cross-platform, support for Unix/Linux, *BSD, eCos, Windows, OS X, QNX, and more.
- CGI, SSI, Digest (MD5) authorization, WebSocket, and WebDAV support
- Resumed download, URL rewriting support, and HTTP proxy support
- SSL support, both one-way and two-way SSL
- IP address-based ACL, Windows service, GET, POST, HEAD, PUT, and DELETE methods

How To Install Mongoose On FreeBSD?

To install mongoose from the ports mechanism:

```
# cd /usr/ports/www/mongoose

# make install clean
```

To install mongoose with the package manager:

```
#pkg install mongoose
```

You can start mongoose at boot time by:

```
# sysrc mongoose_enable="YES"
```

If you restart your machine, mongoose web server will serve /var as http file sharing on port 8080. You can see contents of /var by browsing 127.0.01:8080:

```
# curl 127.0.0.1:8080
```

And just to make sure that mongoose is up and running, issue the following command:

```
# /usr/local/etc/rc.d/mongoose status
```

Output:

mongoose is running as pid 4218.

Or you can find it by listening port:

```
# sockstat -4l
```

Output:

USER COMMAND PID FD PROTO LOCAL ADDRESS FOREIGN ADDRESS

root mongoose 4218 5 tcp4 *:8080 *:*

mongoose does not detach from terminal, and it uses current working directory as the web root, unless -r option is specified. It is possible to specify multiple ports to listen on. For example, to make mongoose listen on HTTP port 80 and HTTPS port 443, one should start it as: mongoose -s cert.pem -p 80,443s

Unlike other web servers, mongoose does not require CGI scripts to be put in a special directory. CGI scripts can be placed anywhere.

Serving A Web Site With Mongoose

First, you can create a simple html file. Let's call this file index.html , and put it on /usr/local/www

```
# mkdir -p /usr/local/www

# cd /usr/local/www

# ee index.html
```

and put this on index.html :

```
<!DOCTYPE html>

<html>

<body>

<h1> BSDMAGAZINE </h1>

<p>  bsdmag.org </p>

</body>

</html>
```

Then, run mongoose by typing the following:

```
# mongoose -listening_port 127.0.0.1:80
```

Mongoose will listen on localhost port 80. If you have many other interfaces, you can bind mongoose to a specific interface.

Disable Directory Listing

You can disable directory listing by typing the following:

```
# mongoose -listening_port 127.0.0.1:80
-enable_directory_listing no
```

Log Access To Website

This command will log all access to log.txt at the same path as index.html:

```
# mongoose -listening_port 127.0.0.1:80
-access_log_file log.txt
```

The logs look like this:

```
127.0.0.1 - - [19/Nov/2017:20:37:49
+0330] "GET / HTTP/1.1" 304 0 -
"Mozilla/5.0 (X11; FreeBSD amd64;
rv:56.0) Gecko/20100101 Firefox/56.0"
```

How To Secure Mongoose Web Server?

There are so many tuning we can add to mongoose, but two of them are necessary:

Change running user to www

```
mongoose -listening_port 127.0.0.1:80
-access_log_file log.txt -run_as_user
www
```

If mongoose crashes, only mongoose will go down not the entire server.

Change www permissions to proper value

```
chmod -R -w /usr/local/www
```

This command removes write permission so a hacker can't run shell on your server.

Change www folder owner

chown -R www:www /usr/local/www

Only www can add or remove content to this folder.

Access Control List

```
mongoose -listening_port 192.168.1.1:80
-run_as_user www -access_control_list
-0.0.0.0/0,+192.168.3.0/24
```

This command runs mongoose on 192.168.1.1 port 80, and denies connections from everywhere, except for 192.168.3.1/24.

Tip: we can't call this firewall but you can do some tricks.

Conclusion

Internet of things (IoT) is getting more popular, and maybe FreeBSD and Mongoose will be a wise choice. With FreeBSD and Mongoose, you can run a full-fledged, fast and minimal web server. Additionally, you can run mongoose on non-embedded devices. For example, "corebox.ir" is based on mongoose web server.

Useful Links

https://github.com/GerHobbelt/civet-webserver/wiki/Mongoose-Manual

https://linux.die.net/man/1/mongoose

http://in4bsd.com

http://meetbsd.ir

Meet the Author

Abdorrahman Homaei has been working as a software developer since 2000. He has used FreeBSD for more than ten years. He became involved with the meetBSD dot ir and performed serious training on FreeBSD. He started his company, etesal amne sara tehran, in February, 2017. His company is based in Iran Silicon Valley.

Full CV: **http://in4bsd.com**

His company: **http://corebox.ir**

Using PostgreSQL Foreign Data Wrapper to Keep Track of Files

You will learn ...

- How to Use PostgreSQL Foreign Data Wrapper to Keep Track of Files
- Compiling and Installing the Foreign Data Wrapper
- How to Create the Extension
- How to Create the File System Table
- Creating a Snapshot of Files

In this paper, you will see how PostgreSQL can be *extended* to pull data out of *special /data sources* that allow the database cluster to query the outside world called *Foreign Data Wrapper/s.* There are many implementations of FDW that allow PostgreSQL to live-query other databases, as well as other data sources like web pages, files, processes, and so on.

This paper proposes a simple setup of a *File System FDW* that allows a system administrator or an application to query the filesystem to get information about files, as well as storing at least one historical version of the latter. The approach presented here is not meant, to any extent, to substitute the traditional and better suited *Source Control Management* software (like RCS and alike). Moreover, all the examples provided aim only to present the reader with a simple background on the capabilities that FDW allow.

Introduction

Imagine you want to store some information about your system configuration file in a database. The solution is straightforward: build an application that can perform some *DML (Data Manipulation Language)* against a database.

Another approach is to use a *File System FDW*. A layer that connects your database directly to a *File System Data Source* so that instead of the database waiting for new data to be stored, it can (to some extent) pull the data automatically.

In this article, I will show you how to use the *Multicorn* FDW to achieve a *poor-man database-SCM*.

To execute the code snippets, you need:

- git and gmake installed;
- python version 2.7 or higher;
- PostgreSQL (a recent version, for this article, I used version 9.6.5);
- Access to privileged user capabilities (e.g., using sudo).

You will also need some basic knowledge about PostgreSQL, how to create a database, a superuser role, and so on. You can get more information reading the online documentation or my previous articles on the matter.

Compiling and Installing the Foreign Data Wrapper

There are several Foreign Data Wrappers (*FDW*) available for PostgreSQL. In this example, we are going to use the *Multicorn* FDW, a set of Python modules that provide several FDW implementations within the same installation. One of such implementation is the *File System FDW*.

The first step is to get the latest *Multicorn* implementation. In this example, you will install the development version obtained via *Git*:

```
% git clone
git://github.com/Kozea/Multicorn.git
```

Before you can actually compile *Multicorn*, you need to adjust it to compile on FreeBSD:

Edit the preflight-chech.sh file, and change the first line with the current available *Bash*, that is:

```
% head -n1 preflight-check.sh

#!/usr/local/bin/bash
```

Remember to run gmake instead of make, so:

- Create the Extension

To create the extension, you need to connect to the PostgreSQL database as superuser, and then load the *Multicorn* extension. After that, you need to define a *Data Server*, an entry point for external data to come into the database.

Therefore:

```
# CREATE EXTENSION multicorn;

# CREATE SERVER filesystem_server

  FOREIGN DATA WRAPPER multicorn
  OPTIONS ( wrapper
'multicorn.fsfdw.FilesystemFdw' );
```

- Create the File System Table

Suppose we want to collect information about the /usr/local/etc/ configuration files. Therefore, you need to define a table that will contain various data:

- the *filename*;
- the *content* (as text);

We can elaborate a little more by adding a *hash* column, and the date the file has been *inspected*.

Therefore, the table will be defined as:

```
# CREATE FOREIGN TABLE usr_local_etc (

    full_file_name text,

    content        text,

    service        text

) SERVER filesystem_server

  OPTIONS( root_dir '/usr/local/etc',

           pattern '{service}.conf',

           content_column 'content',

           filename_column 'full_file_name' );
```

Now, you can try it with a simple SELECT statement:

```
# SELECT service, full_file_name
  FROM usr_local_etc;

 service  | full_file_name
----------+----------------
 pkg      | pkg.conf
 tcsd     | tcsd.conf
 pcp      | pcp.conf
 pgpool   | pgpool.conf
 pool_hba | pool_hba.conf
 idn      | idn.conf
 idnalias | idnalias.conf
```

However, there is a hidden problem: while the user can run simple *stat* commands on the filesystem,

he/she cannot get the content of the files. In fact, if you try to get the content of a file you'll get an error:

```
# SELECT service, content FROM usr_local_etc;

ERROR:  Error in python: OSError

DETAIL:  [Errno 13] Permission denied:
'/usr/local/etc/tcsd.conf'
```

The problem arises from the fact that /usr/local/etc/tcsd.conf has no world-readable flag. A quick solution is to allow another user to read by either changing the file mode (e.g., 644) or to invite the user running the PostgreSQL server to the group of the file owner (in this case _tss), and setting the mode to 640.

```
% id postgres

uid=770(postgres) gid=770(postgres)
groups=770(postgres)


% sudo pw usermod -n postgres -G _tss


% id postgres

uid=770(postgres) gid=770(postgres)
groups=770(postgres),601(_tss)
```

Once the above problem is solved and the trick applied to any problematic file, you can query the table to get living data from the underlying file system (See Listing 1).

Listing 1. Living data

```
# SELECT service, content
  FROM usr_local_etc
  WHERE service = 'pkg';
 service |                                  content
---------+-------------------------------------------------------------------
 pkg     | # System-wide configuration file for pkg(8)                       +
         | # For more information on the file format and                     +
         | # options please refer to the pkg.conf(5) man page                +
         |                                                                   +
         | # Note: you don't need to have a pkg.conf file.  Many installations+
         | # will work well with no pkg.conf at all or with an empty pkg.conf +
         | # (other than comment lines).  You can also override any of these  +
         | # settings from the environment.                                  +
         |                                                                   +
         | # Configuration options -- default values.                        +
         |                                                                   +
         | #PKG_DBDIR = "/var/db/pkg";                                       +
         | #PKG_CACHEDIR = "/var/cache/pkg";                                 +
...
```

Creating a Snapshot of files

Using the Foreign Data Wrapper, the database will *query* the filesystem each time you issue a query, and this means the data in the usr_local_etc table will change accordingly to changes performed outside the database.

If you need to keep a *snapshot* of the file content, let's say to implement a *poor-man* file control management, you can use a *materialized view*.

A materialized view is a *view* over data that is populated by a snapshot of data pulled out from a table. Each time you refresh the view, new data is pulled out of the table. Otherwise, the view will provide a static snapshot of the data at the time it was last updated.

To better explain it, let's create a materialized view to get the content of the files into the file system:

```
# CREATE MATERIALIZED VIEW
usr_local_etc_snapshot AS

    SELECT service, full_file_name, content,
     current_timestamp AS ts,

     md5( content ) AS hash

  FROM usr_local_etc
    ORDER BY service

    WITH NO DATA;
```

When you decide to pull updated data from the filesystem into your snapshot, do the following:

```
# REFRESH MATERIALIZED VIEW
usr_local_etc_snapshot;
```

Let's check that the data into the view is coherent with what is in the database (See Listing 2.)

Also, check the *MD5* outside of the database:

```
% sudo md5 /usr/local/etc/pkg.conf
~

MD5 (/usr/local/etc/pkg.conf) =
84925257b233f69068214cdaf3f630a2
```

As you can see, the *MD5* is the same. Therefore, the data in the materialized view does represent the current snapshot of the filesystem.

Now, imagine you modify the pkg.conf file so that it is updated outside of the database:

```
% sudo emacs /usr/local/etc/pkg.conf
...

% sudo md5 /usr/local/etc/pkg.conf

MD5 (/usr/local/etc/pkg.conf) =
a82431a939e221dd5fc8b702542a30d4
```

Listing 2. Database

```
# SELECT full_file_name, hash, ts

  FROM usr_local_etc_snapshot;


 full_file_name |               hash               |              ts
----------------+----------------------------------+-------------------------------

 pkg.conf       | 84925257b233f69068214cdaf3f630a2 | 2017-11-09 16:54:30.668574+01

...
```

Listing 3. Reports

```
# SELECT full_file_name, hash, ts
  FROM usr_local_etc_snapshot
  WHERE service = 'pkg';


 full_file_name |               hash               |              ts
----------------+----------------------------------+-------------------------------
 pkg.conf       | 84925257b233f69068214cdaf3f630a2 | 2017-11-09 16:54:30.668574+01
```

Then, let's see what the materialized view reports (See Listing 3 above).

As expected, **it does still report the old hash**, the data within the materialized view which has not been modified. What this means is that the content column of the view also has a track of the old (i.e., before editing) content of the same file, allowing for a quick (and dirty) restore of the file content.

What has Changed?

The fact that the materialized view contains the snapshot of the filesystem allows for querying the status of the filesystem itself against the previous (last) snapshot:

```
# WITH current AS (
    SELECT service, md5( content ) AS hash
    FROM usr_local_etc
  )
 SELECT service, ts AS ModifiedSince
    FROM usr_local_etc_snapshot snapshot
    WHERE snapshot.hash <> (
                SELECT hash
                FROM current
                WHERE service =
snapshot.service )
 UNION

 SELECT service, ts AS ModifiedSince
    FROM usr_local_etc_snapshot snapshot
    WHERE NOT EXISTS (
                SELECT service
                FROM current
                WHERE service =
snapshot.service );
```

Meet the Author

Luca Ferrari lives in Italy with his beautiful wife, his great son, and two female cats. Computer science passionate since the Commodore 64 era, he holds a master degree and a PhD in Computer Science. He is a PostgreSQL enthusiast, a Perl lover, an Operating System passionate, a UNIX fan, and performs as much tasks as possible within Emacs. He considers the Open Source the only truly sane way of interacting with software and services. His website is available at http://fluca1978.github.io

The above query is made up of three parts:

- current is a CTE (*Common Table Expression*), a sub-query that computes the hash on the current file system data (i.e., querying the FDW);
- the first SELECT extracts all files that have been modified since the last snapshot (i.e., since the last REFRESH MATERIALIZED VIEW);

the second SELECT extracts all files deleted since the last snapshot.

Running the above query provides the following result:

```
 service |          modifiedsince
---------+-------------------------------
 pkg     | 2017-11-09 16:54:30.668574+01
```

meaning that the pkg service has been modified since the last time it was taken into the materialized view.

Conclusions

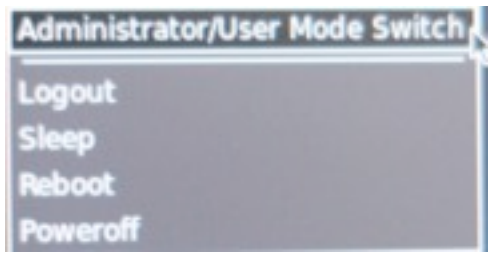
This article has demonstrated a concrete application of PostgreSQL Foreign Data Wrappers feature to allow the database to query other data sources, in particular, a file system to get and track file information. There are a lot of FDW implementations allowing even more, like web browsing and parsing, other database querying, web service interactions and so on. These can all be used as building blocks for a more complex layer of data management.

References

PostgreSQL web site: http://www.postgresql.org
PostgreSQL FDW: https://wiki.postgresql.org/wiki/Foreign_data_wrappers
Multicorn FDW: http://multicorn.org/

Free RDP Configuration

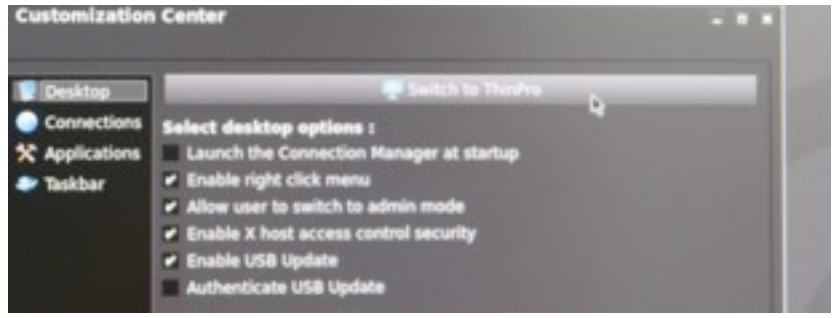
In this article, you will learn how to setup HP t620 Thin Client with Linux Kernel. First, we must enable the Admin Mode. To do this right-click anywhere on the desktop. Then, click on “Switch Admin/User Mode”.



If this is your first time, you must set a password which you will use to access the Admin Mode. As soon as you are logged in, a red border will appear around the desktop to signalize that you’re in the Admin Mode.

Now, we can start with the configuration. I suggest switching to the ThinPro OS. To do this, we have to access the settings through the taskbar, hover over “Setup” , and then click on “Customization Center”.

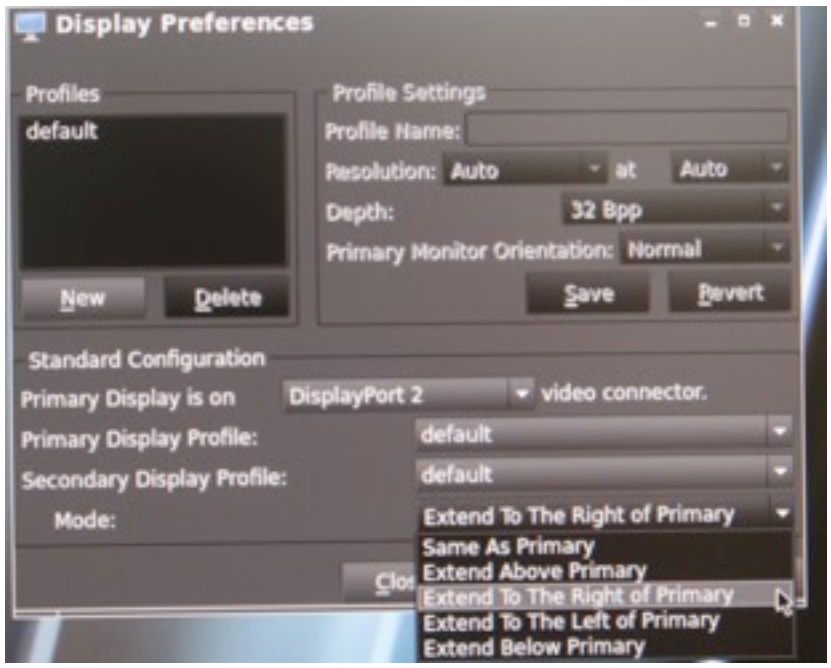
In the window that has just opened, click on the “Switch to ThinPro” button and allow the ThinPro OS to load. This shouldn’t take too long. When the OS has finished loading, proceed with the configuration.



Displays

To edit our display settings, let’s click on the settings button in the taskbar. Then, hover over “Peripherals” and click on “Display Preferences”.

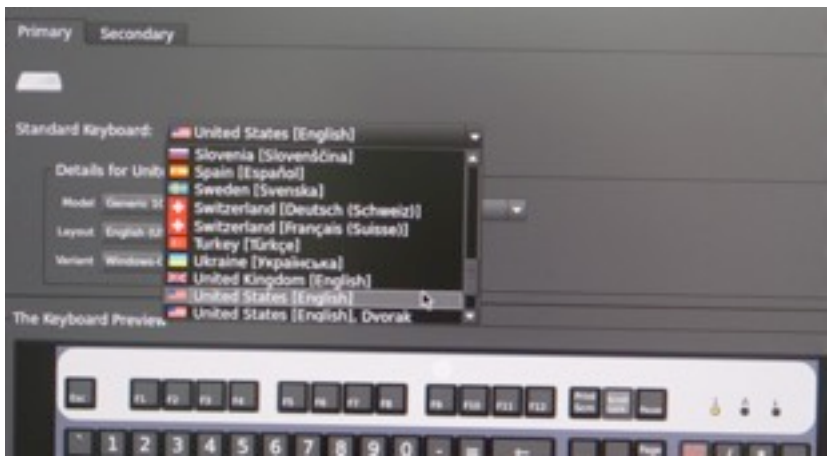
At this stage, we can choose which display to be the primary display and which one to be the secondary display. Also, we can choose the direction in which the screen should extend.



Keyboard

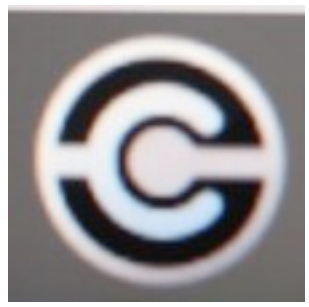
By default, the keyboard layout is set to US (United States). To change the keyboard layout, right-click on the “US” letters at the bottom right of the primary display. Then, click on “Keyboard layout”

In the active window, choose your preferred keyboard layout in the “Standard Keyboard” dropdown.

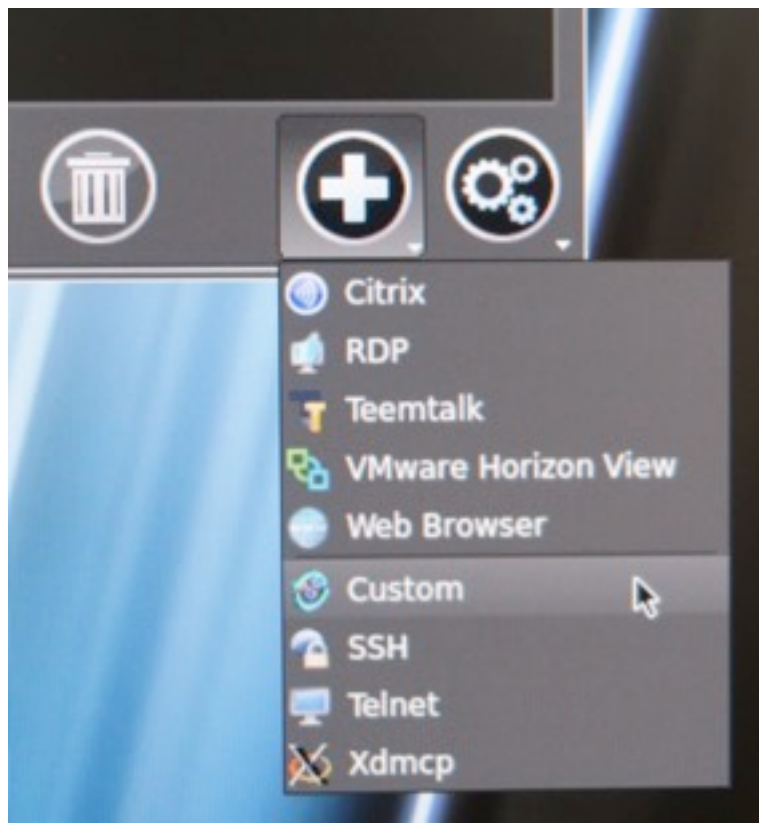


Setting up a remote desktop connection

To start a remote desktop connection, we must open the connection manager. We can open it by clicking on the following symbol in the taskbar:



The next step is deciding what kind of remote connection we would like to establish. For this example, let’s choose a custom connection.



Let’s give the connection a good name, so that we know to which PC to connect. I recommend using other PCs’ IP as the name of the connection. Let’s start with the command we want to run. First, we must decide which program we’re using. Let’s choose xfreerdp, and type it in the command box accordingly. Now we could just enter the IP of the PC we wish to connect to and start the connection, but if we do that, the resolution will be quite uncomfortable to look at and only one monitor will be used. That’s not necessarily bad if we only want to use one monitor for the remote connection. However, if we want to use more, we must add some commands. An

important thing to mention is that the order of the commands we enter matters. That's why we'll start by fixing the resolution.

Resolution

To get rid of the nasty resolution, type the following commands right after the "xfreerdp" in the command box:

+aero - This will manage the desktop composition of the remote connection.

+smart-sizing - Scales the remote desktop to the window size.

+fonts - Enables smooth fonts, this will make the resolution much more comfortable.

-f - Full screen mode

The following commands are used to decide the resolution of the window:

/monitors:0,1 This will determine how many monitors will be used. Make sure to start counting from zero.

/multimon This enables you to use multiple monitors.

/w: Determines the width of the window.

/h: Determines the height of the window.

/size: Determines the screen size. (<width>x<height>)

/compression:off Disables compression

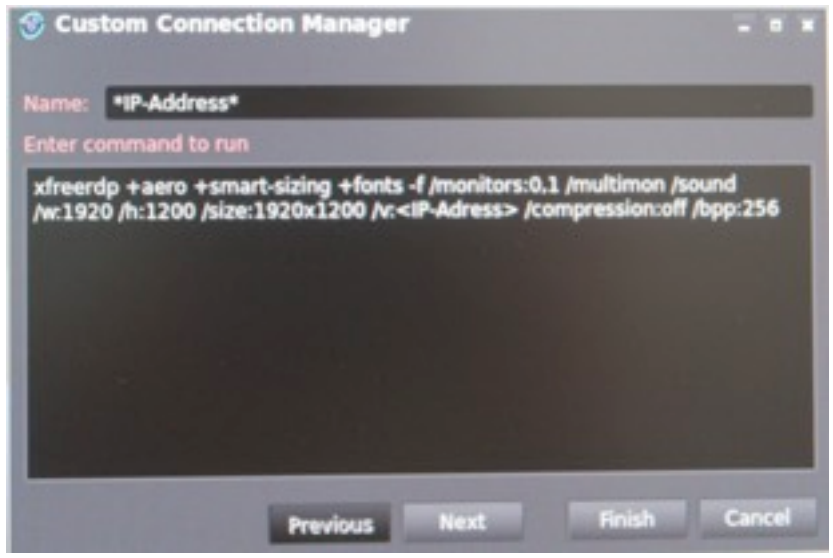
/bpp: Defines the color depth

Remaining commands:

/sound Enables sound from the connection.

/v: *IP-Address* Here, we must enter the IP of the device we want to connect to.

In the end, the complete command should look something like this:



Meet the Author

Loris Zimmerman is an IT student who works at OBRO AG in Switzerland. He is always interested in computers and related stuff. He started working in IT one and a half years ago. If you wish to contact him, send an e-mail to: loris.zim@hotmail.com

Among clouds Performance and Reliability is critical

Download syslog-ng Premium Edition product evaluation here

Attend to a free logging tech webinar here

BalaBit IT Security

www.balabit.com

syslog-ng log server

The world's first High-Speed Reliable Logging™ technology

HIGH-SPEED RELIABLE LOGGING

- above 500 000 messages per second
- zero message loss due to the Reliable Log Transfer Protocol™
- trusted log transfer and storage

Interview with Abdorrahman Homaei

Can you tell our readers about yourself and your role nowadays?

Currently, I am busy with daily administration tasks and CoreBOX development which are getting harder and intense. Besides, I have a company located in Iran Silicon Valley and have to manage my enterprise.

How you first got involved with programming and the FreeBSD world?

About 12 years ago, I was an active and professional windows developer. Secure programming was what introduced me to the FreeBSD world. On FreeBSD, everything was orderly and documented. I think FreeBSD is the developer's paradise. I wrote my first application on FreeBSD 12 years ago, and as you can imagine, there was no headache like windows, no undocumented API, and no crashing.

While having a wide field of expertise, please tell our readers on which area you put the most emphasis, and why?

In my view, security is the most important expertise irrespective of the OS you are using. It doesn't matter how hard you interact with that OS, with shell or with 3D GUI, or who you are, if someone can hack you, then your business is not reliable and you are the loser. Hence, I put much emphasis on security since it is the most critical area.

What was your best work? What was the idea behind it? What was its purpose?

My best work was migrating my desktop to FreeBSD. Using FreeBSD as desktop is so complicated. Every day, you face serious challenges but after a while, you will learn everything and become a geek. Using FreeBSD as desktop teaches you how to solve any problem.

What is your the most interesting programming issue you have encountered, and why?

Migrating to FreeBSD was not easy. I was a device driver developer, but when you migrate to other OS(FreeBSD) and you cannot even work with its command line, it's so hard to develop a simple application. Therefore, programming a device driver was impossible.

What tools do you use most often, and why?

CSH is my best friend because I can do everything in shell and it gives me a good feeling. I also frequently use shell utilities like SSH. When it comes to development, I use C++ , QT , and many more.

What was the most difficult and challenging implementation you've done so far? Could you give us some details?

I think you are talking about CoreBOX exactly. CoreBOX has a brilliant idea behind. Using FreeBSD as a role-based hypervisor is state of the art. In the beginning, you must choose your mechanism to control the hypervisor. You can create a web-based access or application access like many others. Selecting each one will force you to learn how to authenticate users, send and receive data.

CoreBOX neither uses web-based access nor a custom application. CoreBOX is clientless, and you can connect to virtual desktop.

Can you tell us more about your company?

My company's name is "etesal amne sara tehran". I have a 5 year old daughter, and I named the company after her name, Sara. My company is based in Iran Silicon Valley. Our main domain is virtualization, and we use FreeBSD as our infrastructure.

What is CoreBOX?

CoreBOX is a Type-2 FreeBSD-Based High-Performance hypervisor, designed for building carrier-grade virtual infrastructure.

What future do you see for FreeBSD and other OSes? Can you tell us about your favorite features in the new releases?

It seems FreeBSD is more focused on single-board computers. Many companies like FreeBSD because of its liberal license. I hope we will see more from FreeBSD and NetBSD in IoT market. Support for the Allwinner A13 board has been added, and it's interesting.

Do you have any specific goals for the rest of this year?

My goal is to add CoreBOX new features like adding new resource scheduler and auto-tuning.

What's the best advice you can give to the BSD magazine readers?

FreeBSD is an enterprise-class operating system which is reliable and secure. The only way to learn FreeBSD is to install it on your desktop.

Thank you

Interview with Oleksandr Tymoshenko

Can you tell our readers about yourself and your role nowadays?

My name is Oleksandr Tymoshenko. I am a software developer with more than 15 years of experience. Over these years, I worked on a number of projects in various fields including Linux PDA software, SMS center for GSM telco, servers for multiplayer games, IP PBX box, and firmware for VoIP phones.

I am have been a FreeBSD committer since 2008 when I started as a FreeBSD/MIPS developer, but switched camp to FreeBSD/ARM sometime later.

At the moment, I work for Dolby Laboratories as a senior software developer, building conferencing products. We don't use FreeBSD in our products, FreeBSD is just my hobby.

How you first got involved with programming and the FreeBSD world?

I was fascinated by computers between theage of 8 to 9, but only got a chance to work with them when I switched school at 14. I started learning Turbo Pascal, then 8086/80286 assembly and couple years later, after I had got access to local university's HPUX system, I picked up some basic C knowledge. In my second year at the university, I got a job at university's network operations center. They used FreeBSD on most of the systems, and that was the first time I tried this OS. I think it was FreeBSD 3.0. I did some sysadmin work for NOC, experimented with kernel hacking, just examples mostly, nothing really exciting.

My first commercial FreeBSD experience was porting drivers for telephony cards (PCI boards you could connect to phone lines) for small IP PBX startup. While working for this startup, I came across MIPS boards and got interested in its architecture. I thought that it would be nice to run FreeBSD on them. So I found some initial work in this area done by Juli Mallet, and started experimenting with things. That was the start of my active contribution to a FreeBSD project.

While having a wide field of expertise, please tell our readers on which area you put most emphasis, and why?

Since I don't base my career on FreeBSD work, I pick whatever is fun to toy with. I like working on hobbyist ARM boards like Raspberry Pi and Beaglebone products families. Hardware-wise, they are simple enough so you can easily master thewhole architecture. There are no complex clock domains or super-intricate power management controls, and yet they're powerful enough so you don't have to fight for every kilobyte of RAM. You can easily extend them with external devices using I2C, GPIO or SPI buses. They're pure tinkering material and a lot of fun.

What was your the best work? What was the idea behind it? What was its purpose?

There is no single project I can point at and call it a magnum opus. I'd say the cumulative contributions to FreeBSD is my best work so far. At least most impactful. They're building blocks and stepping stones for other people's projects. It's very rewarding to see your work being used by other developers and hobbyists in most unexpected ways. Or how once buggy and unstable code after some time and effort becomes a rock-solid platform for someone else's product.

What is your the most interesting programming issue you've encountered, and why?

To be honest, I don't have any interesting debugging war stories. Debugging is exciting in a puzzle-solving way during the process. However, even when hours of painstaking search boils down to one-liner fix: missing cache sync operation, memory barrier or in worst case, an extra semicolon in a wrong place. It's only entertaining for a day or so.

What tools do you use most often, and why?

For day to day work, I mostly use tmux + vim with few plugins as dev environment, and ack (textproc/ack) for code search. tmux offers powerful features for organizing workspace. I group windows in sessions by theme, i.e., ARM work in progress session with build shell, serial terminal and editor, then another session for Bugzilla work. This environment runs either on either a server or desktop machine, and I can SSH to it and reconnect to the session from anywhere. On my laptop, I use i3, tile-WM that goes well with tmux/vim combination. I use subversion for FreeBSD stuff, git for personal projects, and Perforce at work. I'm a long-time fan of mutt mail client which I use for most of my personal and open-source related communications. Communication software: irssi for IRC and profanity as irssi-like Jabber client.

What was the most difficult and challenging implementation you've done so far? Could you give us some details?

That would be porting u-boot and FreeBSD to Raspberry Pi. I wanted to create FreeBSD port for Pi but to work on it, I needed a way to netboot device. It's essential when you start to work on embedded device support for FreeBSD. Board support is added to kernel config step by step. First, you check if control is passed to kernel entry point by printing something to serial console in _start method. Thereafter, you move debug output further into kernel initialization routine and so on. Sometimes this process is smooth and requires just a few iterations, but often it's multiple fix/build/boot cycles. Without netboot, you have to extract SD card, write new kernel to it, put it back, powercycle board, and check results. Wash, rinse, and repeat. It is slow, dull, and tedious. With netboot, all you need is to build the kernel, copy it to TFTP server, and powercycle board. Board will get address by DHCP, download kernel from TFTP server, and pass control to it. Normally, boards come with versatile boot loader called u-boot (which is de-facto standard these days) built by hardware vendors like Freescale or Marvell. Raspberry Pi back then didn't have it, and proprietary boot code from Broadcomm could only load a kernel from SD card. So I had to port u-boot to Rasperry Pi first using that slow and tedious process. There was no way around it. For netboot, I needed a network card driver. Luckily, u-boot had it. However, I hit a snag beacsue USB controller driverwas absent. Additionally,there was no datasheet and no way to put USB protocol analyzer between USB host and ethernet controller

to see if my changes affect anything. They were both integrated on a board. Hence, I used Linux driver as a reference and tried to implement a minimal subset of functionality to get ethernet working. Every iteration involved exchanging SD card between Pi and desktop. I almost quit a couple of times but out of sheer stubbornness, I kept experimenting and finally was able to transfer the kernel from TFTP server. The code was terrible, but it was good enough to unblock my FreeBSD work.

Can you tell us more about the Bugmeister team? What did you do there?

Bugmeister team works on keeping bugtracker system running, and on making bug reporting and tracking as easy as possible. The less effort required to work on bugs, the higher the chances they'll be worked on. FreeBSD used to use GNATS bugtracking system, much dated and not super flexible to put it mildly. 4-5 years ago, it was decided that the project should switch to a better tool and the Bugmeister team was responsible for picking and deploying it. I was one of the developers working on that task. I resigned from the team in 2014, few months before the actual switch happened. I volunteered back in 2016 to do some routine administrative tasks like helping users with passwords, killing spam PRs, and making minor modifications to Bugzila codebase.

FreeBSD uses a customized version of Bugzilla. The Bugmeister team is responsible for maintaining this version and adding new features to automate some of the tedious work or adopt it to FreeBSD workflows.

Can you tell our readers more about your commits to FreeBSD?

I do not commit to FreeBSD much these days. I am busy with a daytime job, and there is just not enough spare time to get back into FreeBSD flow. I do occasional fixes for some FreeBSD/ARM drivers, commit patches submitted by contributors. But nothing major, unfortunately.

What future do you see for FreeBSD and other OSes? Can you tell us about your favorite features in the new releases?

I am terrible at making predictions. I think FreeBSD is gaining momentum as "true" server/development UNIX. With systemd haunting major Linux distros, people start looking for alternatives and since FreeBSD has more conservative approaches and features like ZFS, it makes a good candidate in server space. But again, this might just as well be my echo chamber.

Moreover, there is ongoing effort to use FreeBSD as an educational OS which I support with all my heart. I think FreeBSD is an example of good engineering and an excellent primer in OS design.

I use 12-CURRENT on my laptop, so I don't wait for new releases. Most exciting recent feature for me was drm-next-kmod port. Now I don't have to build custom kernel branch to get decent Xorg performance from my Kabylake-based Thinkpad.

Do you have any specific goals for the rest of this year?

I want to add VideoCore interface support for Raspberry Pi 3. Pi 3 is 64-bit device while older Pi's are 32-bit, and some work required to port advanced features like audio or OpenGL. It almost works but as usual, there is some weird bug which requires a long enough stretch of time to sit and work on it.

What's the best advice you can give to the BSD magazine readers?

Use the contributions to open-source projects as learning opportunities. Find an area you're interested in, submit patches, ask for feedback, ask questions, and ask for references to code/papers/books. Try mailing lists, try IRC/Slack channels, and try emailing the author. Sometimes there will be no response, sometimes people will forget to follow up, or you might come across a toxic person at some point. Whichever the case, try not to get discouraged. There are a lot of people in open-source community who are willing to share their knowledge and experience: find them, grow your network, and keep those patches coming.

Thank you

MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

Overriding LIBC Functions

READ ONLINE WWW.BSDMAG.ORG

Automating Vulnerability Scanning with Vuls

Your Information Is Out There Ready for Buying

Processing Data in Parallel Using Multithreading

The TrueNAS Unified Storage X10

Interview with David Mytton

VOL 11 NO 07
ISSUE 07/2017 (95)
ISSN 1898-9144

COLUMN

On October 1st, the Network Enforcement Act took effect in Germany. This creates a legal framework for censorship of the Internet. As more and more governments take the hammer of censorship to content, what are the ramifications for free speech, but more importantly, has the Internet come of age?

ROB SOMERVILLE

As a writer and technologist, I fall into the same category as musicians, artists and philosophers have done throughout the ages. Torn between commercial reality and freedom of expression, I have to continually examine any output to ensure that it treads a fine line between entertainment, education, and offence. More often than not, a lot of what I want to say has to be responsible and truthful, yet removes the core emotion and guttural meaning behind the message. Or to put it another way, bring race, religion, politics or money into the equation, you leave yourself open to criticism and / or censure. Which is fair enough, if you have an all-out bias or level of opinion that firmly places you in the category of bigot, bore or banshee. I would agree that the first two categories deserve a certain degree of civilised opprobrium, the latter much less so.

A banshee, according to Wikipedia, is a female spirit in Irish mythology who heralds the death of a family member, usually by wailing, shrieking, or keening. This is not to be confused with a troll or other general troublemaker, as the wail is not so much to cause irritation to the ears but to signal to the wider community that a tragedy, an injustice has taken place. It is not without significance that women are given a wide tolerance to express grief in any civilised society, as they are often economically the individuals who have to bear the consequences of the death of a child or a partner. In this egalitarian age, I would humbly submit that when pushed to the limits, men are also capable of such expression, albeit with fewer tears. And that is the problem, we have opened a Pandora’s box of communication, where all sides can come to the table, argue, make adjustments, refine their weapons, and come back with a stronger case to batter their “opponents” into submission.

Free speech is an integral part of any civilised society. I am loath to say democratic here, as many individuals who have a valid complaint or axe to grind have often been ignored by the wheels of justice. The Internet is a powerful weapon in this regard, as the hypocrisy and inefficiency of government and politics at large can be exposed for what they are. Given a revelation, a narrative, the only response that the guilty party can respond with is either a barrage of PR flak, an attempt to discredit the author, or worse still. The Internet is populated by lunatics of course, with their conspiracy theories and psychologically damaged rantings.

Some of the most convincing arguments I have ever experienced have been propositioned from the online community, and I have been online long before it became a dot on the political radar. There is something about writing, typing, that engages a different part of the brain. It gives the author enough freedom to self-edit, say things more considered than a one-to-one conversation. In the real world, you might be a 6-foot man with a shaved head, considerable muscle and tattoos, and me, a 5-foot female with brittle bone syndrome. Online, all we have is a handle and accountability to the intelligence services and governments that monitor every byte of traffic. That is a game changer, and the powers that be are scared, really scared. And not just of the trolls and genuine lunatics.

All of this boils down to the rule of Three's. We can work inside the system, outside the system, or consider a different path. And here, I will be controversial. The Internet is spiritual. Unless we engage that side of ourselves, all is naught. We are but cogs in a wheel, slaves to a machine, meaningless chemical factories that are doomed to rot. A lot has been argued about fake news, credibility and the like, but we are now facing a crisis of faith. Where the old era was concerned with faith in God, we are now looking at faith in information and content providers. Faith, like everything else, has been neatly divided, then parceled up and placed in a corresponding container. Rather than judging the actions of individuals, those in control have been bewitched by judging their thoughts and motives – an affront to any human being.

The closing of these doors brings this whole problem into a difficult and sensitive arena – politics. Currently, Russia is under attack from Western media (and indeed government) for being ultimately, a purveyor of "fake news". As far as propaganda is concerned, I think the West has a lot to answer for. All of this gibberish (and I can't think of any better word to describe it) goes back, as far as I can remember as a child, to when the USSR managed to take the wind out of the sails of the USA by placing a working satellite in orbit, and the first unmanned probe on the moon. Bereft of physical manifestation, the only recourse was demonisation and character assassination. The result of losing face was not going to be pretty, and as a result, the cold war continued on for many more years. Who knows what world we would be living in if dialogue, engagement, and discourse had been the result of this technological achievement?

Technology, like people, goes through growth stages. The wonder and innocence has now moved on, and we are left at best with an awkward teenager with spots or at worst, a rich individual going through a midlife crisis. The true players are making their presence felt at the watering hole of power, and no matter how dirty their hooves, legs or bodies are, they desire to bathe in the cool waters where others drink. It is time for a clean water act.

ServerU

Rack-mount networking server

Designed for BSD and Linux Systems

DESIGNED FOR FreeBSD | DESIGNED FOR GNU / Linux opensource | DESIGNED FOR PRO apps FreeBSD Enterprise Appliance | DESIGNED FOR pfSense

Designed. Certified. Supported

Up to **5.5Gbit/s** routing power!

KEY FEATURES

- 6 NICs w/ Intel igb(4) driver w/ bypass
- Hand-picked server chipsets
- Netmap Ready (FreeBSD & pfSense)
- Up to 14 Gigabit expansion ports
- Up to 4x10GbE SFP+ expansion

PERFECT FOR

- BGP & OSPF routing
- Firewall & UTM Security Appliances
- Intrusion Detection & WAF
- CDN & Web Cache / Proxy
- E-mail Server & SMTP Filtering

contactus@serveru.us | www.serveru.us

8001 NW 64th St. Miami, LF 33166 | +1 (305) 421-9956